FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Assignment of master's thesis

Title:	Short-Term Precipitation Forecasting from Satellite Data Using Machine Learning
Student:	Bc. Jiří Pihrt
Supervisor:	Mgr. Petr Šimánek
Study program:	Informatics
Branch / specialization:	Knowledge Engineering
Department:	Department of Applied Mathematics
Validity:	until the end of summer semester 2023/2024

Instructions

Accurate precipitation forecasting is essential for various applications, including agriculture, transportation, and disaster management. Ground-based weather radars are common for forecasting precipitation, but they are not always available, especially in remote or poorly-instrumented areas. Satellite data offers an alternative way to observe and forecast precipitation, particularly for these underserved regions. By using machine learning techniques to predict precipitation from satellite data, it is possible to improve short-term forecasting capabilities and provide valuable information to decision-makers in a range of fields.

The task is as follows:

1. Conduct a literature review to gain a thorough understanding of existing techniques for forecasting precipitation and machine learning, especially using satellite data. This will provide the necessary background information for your work.
2. Develop a clear research question and hypothesis for your study. For example, you could ask "Can machine learning techniques be used to accurately forecast severe rain up to eight hours in advance using lower-resolution satellite radiance images?"
3. Inspect and understand Weather4Cast 2022 competition data. Pre-process the necessary data for your study. Describe the dataset.
4. Train and evaluate a machine learning model on your data. You will need to decide on an appropriate model architecture and evaluation metrics, and use them to assess the

model's performance.
5. Analyze and interpret the results of your study. What did you find out about the effectiveness of machine learning techniques for short-term precipitation forecasting using satellite data?

Electronically approved by Ing. Magda Friedjungová, Ph.D. on 5 January 2023 in Prague.

FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Master’s thesis

Short-Term Precipitation Forecasting from Satellite Data Using Machine Learning

Bc. Jiří Pihrt

Department of Applied Mathematics

Supervisor: Mgr. Petr Šimánek

May 4, 2023

Acknowledgements

I would like to express my sincere thanks to my supervisor Mgr. Petr Šimánek for his leadership during the competition and for providing us with an exceptional work environment. I also thank my colleagues Bc. Rudolf Raevskiy and Ing. Matej Choma for their contributions.

Declaration

In Prague on May 4, 2023 . . . .

Citation of this thesis

Pihrt, Jiří. *Short-Term Precipitation Forecasting from Satellite Data Using Machine Learning.* Master's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2023.

Abstrakt

Geostacionární meteorologické satelity jsou zdrojem globálních a častých pozorování počasí, ale nepozorují přímo srážky. V této práci zkoumáme metody odhadování a předpovídání srážek ze satelitních dat. Cílem této práce je předpovědět až 8 hodin radarových snímků srážek s vysokým rozlišením z multispektrálních satelitních snímků s větším kontextem ale menším rozlišením. Pro tento úkol jsme vyvinuli nový model hlubokého učení s využitím neuronových sítí U-Net a PhyDNet. Nazvali jsme jej *WeatherFusionNet*, protože slučuje tři různé způsoby zpracování satelitních dat; předpovídání budoucích satelitních snímků, odhadnutí srážek ve vstupní sekvenci a přímé použití vstupní sekvence. Pro trénování a vyzkoušení modelu na reálných datech jsme se zúčastnili NeurIPS soutěže *Weather4cast 2022*, která poskytuje prostorově a časově srovnané satelitní snímky a cílová radarová data. WeatherFusionNet dosáhla prvního místa v hlavní části soutěže. Dále jsme experimentovali s několika dalšími modely, zkusili zahrnout statická data do vstupu a porovnali náš model s předpovídáním přímo z radaru.

Klíčová slova předpověď počasí, předpověď srážek, satelitní data, strojové učení, hluboké učení, neuronová síť, rekurentní neuronová síť, konvoluční neuronová síť

Abstract

Geostationary meteorological satellites are a source of global and frequent weather observations, but they do not directly observe precipitation. We research existing methods for inferring and forecasting rainfall from satellite data. The aim of this thesis is to predict high resolution precipitation radar observations up to 8 hours ahead from larger context but lower resolution multi-spectral geostationary satellite images. We develop a novel deep learning model for this task, utilizing the U-Net and PhyDNet neural networks. We name it *WeatherFusionNet*, as it fuses three different ways to process the satellite data; predicting future satellite images, estimating precipitation in the input sequence, and using the input sequence directly. To train and test it on real data, we participate in the NeurIPS *Weather4cast 2022* competition, which provides spatially and temporally aligned satellite imagery and target precipitation radar data. WeatherFusionNet achieved first place in the Core challenge of the competition. We further experiment with several different models, try including static data in the input, and compare our model with a direct radar-to-radar model.

Keywords weather forecasting, precipitation forecasting, satellite data, machine learning, deep learning, neural network, recurrent neural network, convolutional neural network

Contents

List of Figures

List of Tables

Introduction

Precipitation forecasting is a long standing scientific challenge with direct societal impact. The task is suitable for machine learning due to vast amounts of continuously collected data and a rich spatial and temporal structure with long range dependencies. In the field of machine learning, deep neural networks have seen remarkable progress in recent years due to increased amounts of available data, better model architectures and ease of implementation on powerful specialized hardware such as GPUs and TPUs. [1]

Recently, a large amount of deep learning research was devoted to precipitation forecasting from ground based precipitation radar observations. However, ground based radars have limited range, and are not present in many less populated areas in the world. [2]

Geostationary meteorological satellites are a tempting source of global, frequent and uninterrupted weather observations. However, they do not directly observe precipitation. This thesis focuses on methods which infer and forecast precipitation directly from satellite images.

The aim of this thesis is to develop a deep learning model for short-term prediction of precipitation from geostationary satellite images. To train and test it on real data, we participate in the NeurIPS Weather4cast 2022 competition, which provides spatially aligned satellite imagery and target precipitation radar data.

Chapters 1 and 2 present a research of current weather forecasting and deep learning methods. Chapter 3 describes the competition and provided data. In Chapters 4 and 5, we present our implemented approaches and experiments.

Chapter 1

Theoretical background

This chapter briefly explains some of the important concepts and methods necessary for the rest of this thesis.

1.1 Weather forecasting

Deep-learning weather prediction models are often compared to traditional techniques. This section provides a brief overview.

1.1.1 Numerical weather prediction

Typical long-term weather forecasts are generated using numerical weather prediction (NWP) models. These models work by utilizing the laws of physics to simulate the dynamics of the atmosphere and forecast future weather based on current observations. The core concept is solving partial differential equations (PDEs) which govern the dynamics of the atmosphere, with an initial condition (the observed weather), also referred to as an initial value problem. This idea is more than 100 years old, even though there were no computers and hardly any frequent atmospheric observations. NWP has seen substantial advances over the past decades due to improvements in the representation of the physics, an increase in observational data and an exponential growth in computing power. [3, 1]

According to [4], there are seven basic equations that should, in theory, fully describe the dynamics of the atmosphere. However, it is mathematically difficult to solve them analytically, so they need to be solved numerically with spatial and temporal discretizations. The computational and power demands of NWP grow as a power of the resolution of the forecast. This creates a trade-off between the accuracy of the forecast, which requires increasing levels of resolution, and the time required to compute the forecast. [4, 1]

NWP simulations from a given input state are usually deterministic. This contrasts with the chaotic nature of the atmosphere. Even small perturbations in the input state can lead to vastly different predictions, and the accuracy decreases over time. It is also unrealistic to assume that the measurements of the current weather are perfect, due to noise and missing coverage of many regions. This calls for a need to measure uncertainty of the forecasts.

The operationally used solution is to compute a set of Monte Carlo simulations with slightly perturbed initial states, referred to as an ensemble. Depending on the predictability of the current weather, the resulting forecasts will differ to varying degrees, providing an estimate of the uncertainty of the predictions. As shown in Figure 1.1, the ensemble can be used to estimate the probability of weather events at a specific time. [4, 3]

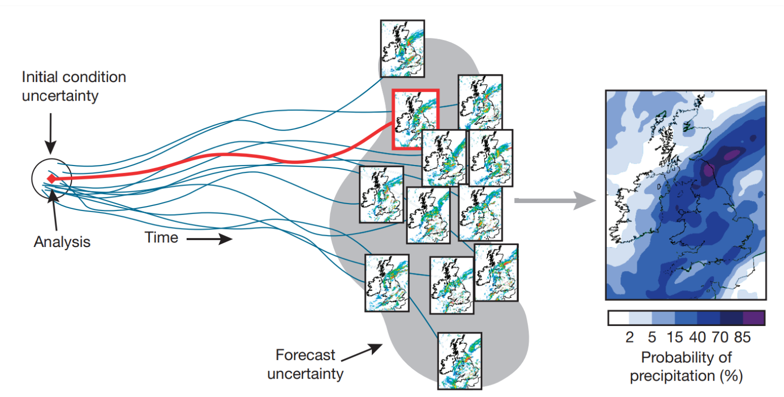

Initial condition uncertainty
Analysis
Time
Forecast uncertainty
2 5 15 40 70 85
Probability of precipitation (%)

Figure 1.1: Visual explanation of estimating the probability of precipitation using a NWP ensemble. [3]

Some of the currently operational NWP systems are:

- GFS (Global Forecast System)[1] and GEFS (Global Ensemble Forecast System)
- ECMWF (European Centre for Medium-Range Weather Forecasts)[2]
- WRF (Weather Research and Forecasting Model)[3]
- HRRR (High Resolution Rapid Refresh)[4]

[1]https://www.ncei.noaa.gov/products/weather-climate-models/global-forecast
[2]https://www.ecmwf.int/
[3]https://www.mmm.ucar.edu/models/wrf
[4]https://rapidrefresh.noaa.gov/hrrr/

1.1.2 Radar nowcasting

NWP methods typically perform long-term forecasts over a large area, but at the cost of a lower spatial and temporal resolution and longer compute time, which makes them less suitable for very short-term forecasting (several hours ahead), often called nowcasting. A desirable source of precipitation measurements are precipitation radars, which produce frequent and low latency observations.

Traditional precipitation nowcasting is usually performed in two steps through the extrapolation of radar images, based on Lagrangian persistence. First, a motion field (or advection field) is estimated from a sequence of past radar images. See Figure 1.2 for an example. The techniques developed for this task in meteorology generally match the optical flow estimation algorithms developed in computer vision, such as the Lucas–Kanade [5] method. In the second step, the most recent radar observation is extrapolated using the estimated motion vectors. [2]

These methods can be roughly divided into two categories:

- Object-based extrapolation first identifies a convective storm cell, and then extrapolates its trajectory based on the estimated motion. This technique is mainly suitable for nowcasting convective storms with high-intensity and stability.

- Region-based extrapolation techniques work directly with the radar image and extrapolate all grid values without specific classifications.

However, these methods work under the assumption that the motion and intensity of the precipitation field are constant. Their performance is poor when forecasting rapidly changing weather, especially for severe storms with abrupt intensity, location and size changes. They are unable to represent the dynamics of storm initiation or decay. [6, 7]

Some extrapolations methods are implemented in open-source Python libraries `rainymotion` [8] and `pySTEPS` [9]. These can serve as reasonable baselines when developing deep-learning models for this task.

1.2 Deep learning model architectures

This section describes the neural network architectures used in the practical part of this thesis. Knowledge of machine learning techniques, especially recurrent and convolutional neural networks, is assumed.

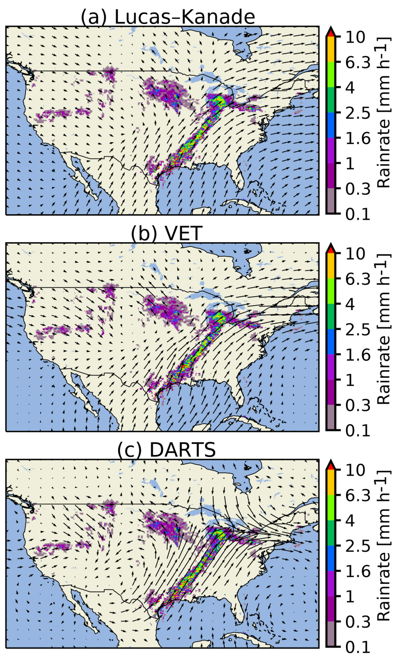


Figure 1.2: Comparison of advection fields obtained by various optical flow methods from the `pySTEPS` library. [9]

1.2.1 U-Net

U-Net [10] is an image-to-image convolutional neural network. The name is inspired by its U-shaped architecture shown in Figure 1.3. It can be described as a symmetrical encoder-decoder architecture, consisting of a contracting path to capture context and a symmetric expanding path that enables precise localization, further enhanced by the usage of skip/shortcut connections.

The original U-Net proposed by [10] is illustrated in Figure 1.3. The contracting path (encoder) follows the typical architecture of a convolutional network, with unpadded convolutional layers followed by a ReLU activation function and max-pooling to downsample the image resolution. At each downsampling step, the amount of feature channels is doubled. Every step in the expansive path begins with a 2×2 up-convolution (transposed convolution) which also halves the amount of feature channels, a concatenation with the output of the corresponding encoder layer, and two more 3×3 unpadded convolutions, each followed by a ReLU. At the final layer, the 64 channel output is mapped to the desired amount of output channels using a 1×1 convolution.

U-Net likely gained a lot of popularity due to its simple architecture, which opens up the door to a lot of possible modifications. Over the years, many improvements and variants have been developed. For example, the unpadded

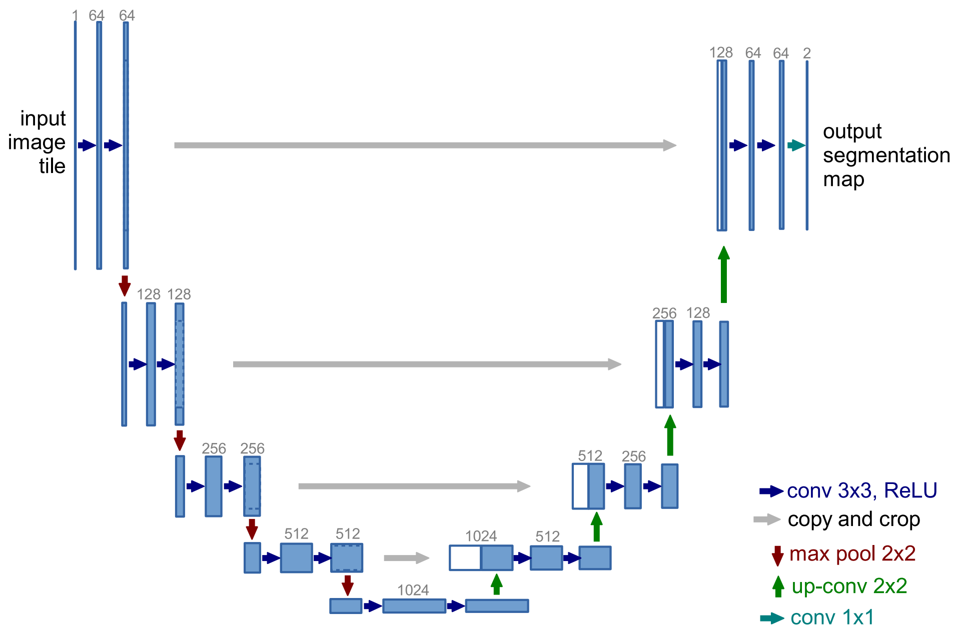


Figure 1.3: The original U-Net architecture. Blue boxes represent multi-channel feature maps. The number of channels is denoted on top of the box. White boxes represent feature maps copied in skip connections. The arrows denote the different operations. [10]

convolutions from the original proposal can be replaced with padded convolutions, enabling the model to output exactly the same dimensions as the input. Many variants make use of batch normalization [11] or similar normalization techniques. U-Net can also be augmented with attention modules [12], or nested skip connections [13]. A 3D U-Net [14] variant replaces 2D convolution layers with 3D convolutions for processing 3D data.

Although U-Net was originally designed for image segmentation (classification of each output pixel), it can be used in various image-to-image translation tasks, such as denoising [15] or super-resolution [16]. Spatio-temporal prediction can be also be approached as an image-to-image translation task, where the temporal dimension is flattened by concatenating the frames in the channel dimension. Compared to recurrent networks, U-Net has a lower computational cost and is more powerful in maintaining the multi-scale spatial information of input data [17]. The contracting and expanding mechanisms in U-Net can propagate information over large distances from input to output images without the need for large convolution kernels, which are computationally expensive. U-Net is widely used in spatio-temporal forecasting tasks including weather forecasting [18, 19, 20, 7]. 3D U-Net can also be used for spatio-temporal prediction [21, 6], where the temporal dimension is considered as the third dimension and also processed with convolutions.

1.2.2 ConvLSTM

Recurrent neural networks (RNN) are commonly used for time series. They are suited for tasks such as classifying all items in a sequence or predicting the next items in a sequence. A successful and widely used RNN is the LSTM (Long Short-Term Memory) [22] neural network.

Although LSTM has proven powerful for handling temporal correlation, it has too much redundancy for spatial data, due to the usage of dense (fully connected) layers. Convolutional LSTM (*ConvLSTM*) [23] replaces those full connections with convolution operations. To achieve this, the hidden state $\mathbf{h}_t$, the memory cell state $\mathbf{c}_t$, and all inputs and outputs of the gates $\mathbf{f}_t$, $\mathbf{i}_t$, $\mathbf{o}_t$ are 3D tensors, consisting of two spatial dimensions and a channel dimension. The key equations of ConvLSTM are:

$$\begin{aligned}
\mathbf{f}_t &= \sigma(W_{xf} * \mathbf{x}_t + W_{hf} * \mathbf{h}_{t-1} + W_{cf} \odot \mathbf{c}_{t-1} + b_f) \\
\mathbf{i}_t &= \sigma(W_{xi} * \mathbf{x}_t + W_{hi} * \mathbf{h}_{t-1} + W_{ci} \odot \mathbf{c}_{t-1} + b_i) \\
\mathbf{g}_t &= \tanh(W_{xg} * \mathbf{x}_t + W_{hg} * \mathbf{h}_{t-1} + b_g) \\
\mathbf{c}_t &= \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \mathbf{g}_t \\
\mathbf{o}_t &= \sigma(W_{xo} * \mathbf{x}_t + W_{ho} * \mathbf{h}_{t-1} + W_{co} \odot \mathbf{c}_t + b_o) \\
\mathbf{h}_t &= \mathbf{o}_t \odot \tanh(\mathbf{c}_t),
\end{aligned} \tag{1.1}$$

where $\mathbf{x}_t$ is the input at time step t, W and b are the network's parameters (weights and biases), σ and tanh are the sigmoid and hyperbolic tangent functions, '$*$' denotes the convolution operator and '$\odot$' denotes the Hadamard product (point-wise multiplication). See Figure 1.4 for a visual explanation.

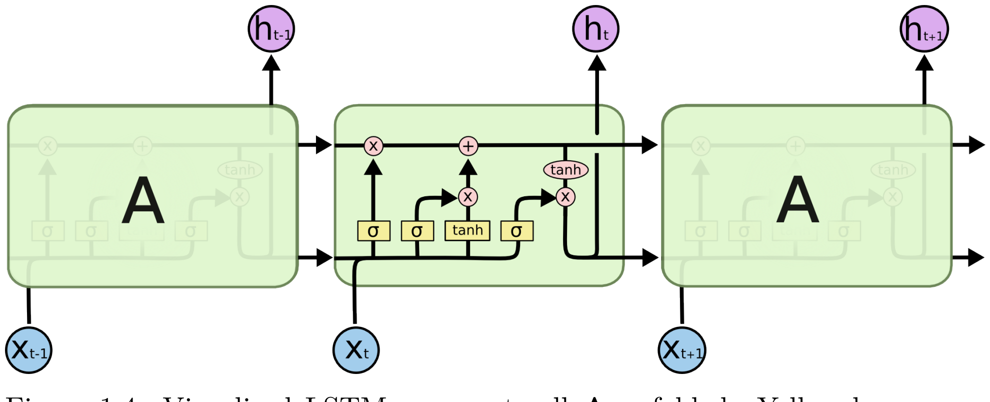


Figure 1.4: Visualized LSTM recurrent cell **A** unfolded. Yellow boxes represent fully connected layers (convolutional layers in ConvLSTM) and corresponding activation functions. [24]

Like the fully connected LSTM, ConvLSTM can also be used as a building block to form more complex architectures. The authors of ConvLSTM propose the architecture shown in Figure 1.5. It consists of an encoding network and a forecasting network, which are both formed from several stacked ConvLSTM layers (cells). The last states and cell outputs of the encoding network layers are copied to the initial states of the forecasting network. The states of the forecasting network are then concatenated and fed into a 1×1 convolutional layer to generate the prediction sequence. [23]

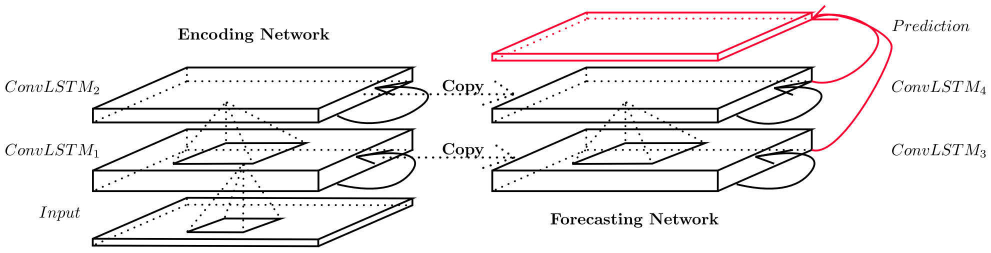


Figure 1.5: Encoding-forecasting ConvLSTM architecture. [23]

1.2.3 PhyDNet

An appealing way to improve spatiotemporal prediction methods is by leveraging physical knowledge described by partial differential equations (PDE). However, physics alone is not enough to describe the full visual content of sequences of images. *PhyDNet* [25] (PhyD stands for physical dynamics) is a deep RNN architecture dedicated to video prediction, which leverages physical dynamics and disentangles it from other complementary information.

To achieve this, [25] propose a disentangling architecture of two branches, as shown in Figure 1.6. The left branch represents the physical dynamics, and features a recurrent physically constrained cell called *PhyCell*, which performs PDE-constrained prediction in latent space. The right branch extracts other residual information required for future prediction, such as visual appearance and details, using ConvLSTM cells. Combining both representations eventually leads to more accurate image sequence prediction. [25, 26]

Formally, let $\mathbf{u} = \mathbf{u}(t, \mathbf{x})$ be the frame of a spatiotemporal sequence at time t for spatial coordinates $\mathbf{x} = (x, y)$, $\mathbf{h}(t, \mathbf{x}) \in \mathcal{H}$ the latent representation of the sequence up to time t, which decomposes as $\mathbf{h} = \mathbf{h^p} + \mathbf{h^r}$, where $\mathbf{h^p}$ represents the physical component and $\mathbf{h^r}$ the residual component of the disentanglement. The sequence evolution in the latent space $\mathcal{H}$ is described by the partial differential equation

$$\frac{\partial \mathbf{h}(t, \mathbf{x})}{\partial t} = \frac{\partial \mathbf{h^p}}{\partial t} + \frac{\partial \mathbf{h^r}}{\partial t} := \mathcal{M}_p(\mathbf{h^p}, \mathbf{u}) + \mathcal{M}_r(\mathbf{h^r}, \mathbf{u}), \tag{1.2}$$

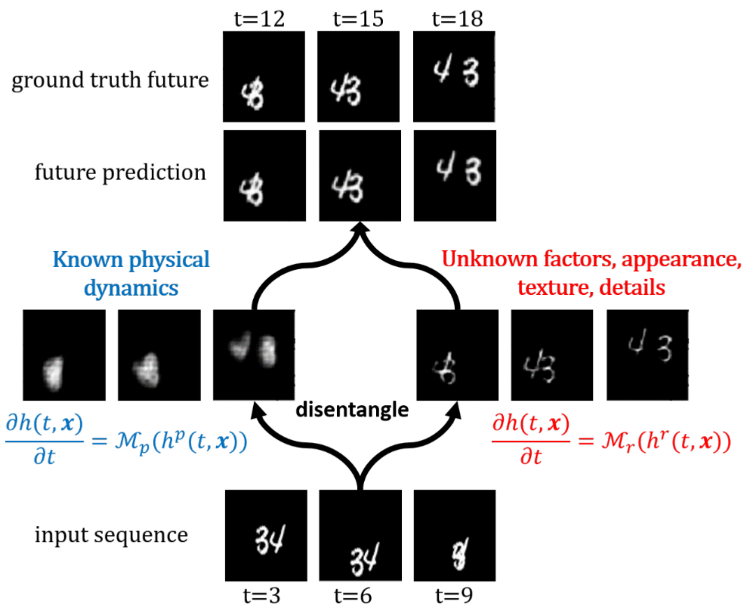

t=12
t=15
t=18
ground truth future
future prediction
Known physical dynamics
Unknown factors, appearance, texture, details
disentangle
$\frac{\partial h(t,\boldsymbol{x})}{\partial t} = \mathcal{M}_p(h^p(t,\boldsymbol{x}))$
$\frac{\partial h(t,\boldsymbol{x})}{\partial t} = \mathcal{M}_r(h^r(t,\boldsymbol{x}))$
input sequence
t=3
t=6
t=9

Figure 1.6: Disentangling architecture of PhyDNet, shown on the Moving MNIST dataset. [25]

where $\mathcal{M}_p(\mathbf{h}^\mathbf{p}, \mathbf{u})$ and $\mathcal{M}_r(\mathbf{h}^\mathbf{r}, \mathbf{u})$ represent physical and residual dynamics in the latent space $\mathcal{H}$. The goal of PhyDNet is to learn the mapping from input sequences to a latent space which approximates these disentangling properties.

A video frame $\mathbf{u}_t$ at time t is mapped by a deep convolutional encoder $\mathbf{E}$ into a latent space representing the targeted space $\mathcal{H}$. $\mathbf{E}(\mathbf{u}_t)$ is then used as input for two parallel RNNs.

The left branch in Figure 1.7 fulfills the physical part of the PDE in Equation 1.2, $\frac{\partial \mathbf{h}^\mathbf{p}(t,\mathbf{x})}{\partial t} = \mathcal{M}_p(\mathbf{h}^\mathbf{p}, \mathbf{u})$. This PDE is modelled by PhyCell, described in the following section (1.2.3.1). This leads to the computation of $\mathbf{h}^\mathbf{p}_{t+1}$ from $\mathbf{E}(\mathbf{u}_t)$ and $\mathbf{h}^\mathbf{p}_t$.

The right branch fulfills the residual part of the equation, $\frac{\partial \mathbf{h}^\mathbf{r}(t,\mathbf{x})}{\partial t} = \mathcal{M}_r(\mathbf{h}^\mathbf{r}, \mathbf{u})$. A generic RNN, such as ConvLSTM, can be used for computing $\mathbf{h}^\mathbf{r}_{t+1}$ from $\mathbf{E}(\mathbf{u}_t)$ and $\mathbf{h}^\mathbf{r}_t$. The network used in [25] consists of three stacked ConvLSTM layers.

The combined representation, $\mathbf{h}_{t+1} = \mathbf{h}^\mathbf{p}_{t+1} + \mathbf{h}^\mathbf{r}_{t+1}$, is then processed by a deep decoder $\mathbf{D}$ to predict the next frame $\hat{\mathbf{u}}_{t+1}$.

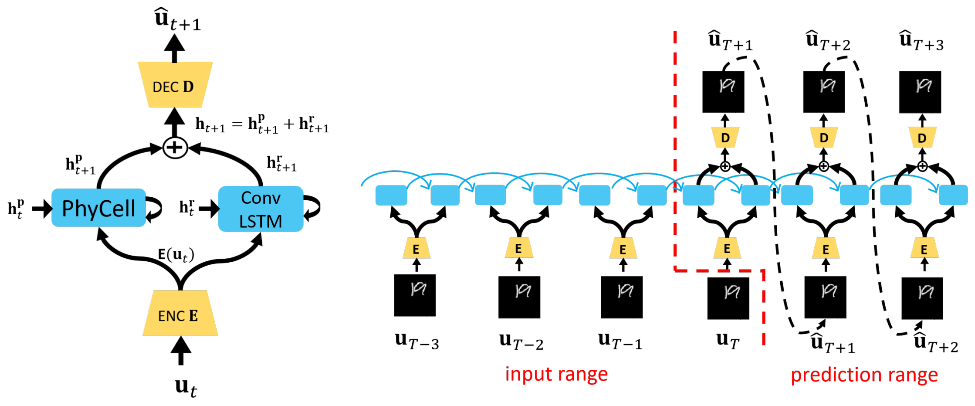


Figure 1.7: Left: PhyDNet recurrent neural network. Right: The same network unfolded, forming a sequence to sequence model suited for multi-step video prediction. [25]

1.2.3.1 PhyCell

PhyCell (Figure 1.8) is a physically constrained RNN cell introduced in [25] that models the dynamics $\mathcal{M}_p(\mathbf{h^p}, \mathbf{u})$ in two steps:

$$\mathcal{M}_p(\mathbf{h}, \mathbf{u}) \coloneqq \Phi(\mathbf{h}) + C(\mathbf{h}, \mathbf{u}). \tag{1.3}$$

The index $\mathbf{p}$ in $\mathbf{h^p}$ is dropped for simplicity. The first step $\Phi(\mathbf{h})$ is a physical predictor modelling the latent dynamics, and $C(\mathbf{h}, \mathbf{u})$ is a correction step, similar to a Kalman filter [27].

The physical predictor $\Phi(\mathbf{h})$ is modelled as follows:

$$\Phi(\mathbf{h}(\mathbf{t}, \mathbf{x})) = \sum_{i+j<q} c_{i,j} \frac{\partial^{i+j}\mathbf{h}}{\partial x^i \partial y^j}(t, \mathbf{x}), \tag{1.4}$$

a linear combination with learned coefficients $c_{i,j}$ of spatial derivatives up to a certain order q. This is implemented with two convolutional layers as shown in Figure 1.8.

The first layer approximates k^2 partial derivatives. The k^2 convolution kernels $\mathbf{w}_{i,j}$ are constrained by moment loss regularization $\mathcal{L}_{\text{moment}}$. Defining a $k \times k$ matrix $\mathbf{\Delta}^k_{i,j}$, which equals 1 at position (i, j) and 0 elsewhere, the regularization term is computed as

$$\mathcal{L}_{\text{moment}} = \sum_{i,j \leq k} ||\mathbf{M}(\mathbf{w}_{i,j}) - \mathbf{\Delta}^k_{i,j}||_F, \tag{1.5}$$

where $|| \cdot ||_F$ is the Frobenius norm and $\mathbf{M}$ is a moment matrix [28]. This regularization term is added to the loss function during training.

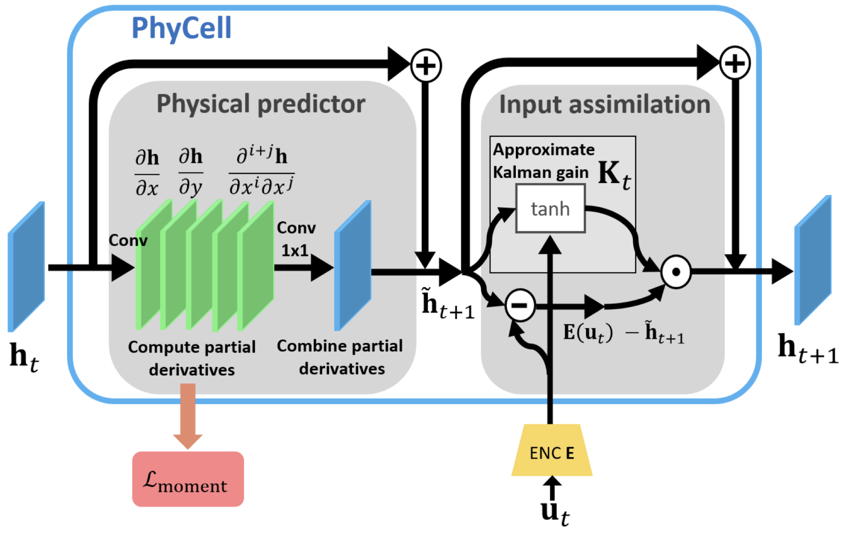

PhyCell
Physical predictor
Input assimilation
$\frac{\partial \mathbf{h}}{\partial x}$ $\frac{\partial \mathbf{h}}{\partial y}$ $\frac{\partial^{i+j}\mathbf{h}}{\partial x^i \partial x^j}$
Conv
Conv 1x1
Compute partial derivatives
Combine partial derivatives
Approximate Kalman gain $\mathbf{K}_t$
tanh
$\mathbf{E}(\mathbf{u}_t) - \tilde{\mathbf{h}}_{t+1}$
$\mathbf{h}_t$
$\tilde{\mathbf{h}}_{t+1}$
$\mathbf{h}_{t+1}$
$\mathcal{L}_{\text{moment}}$
ENC E
$\mathbf{u}_t$

Figure 1.8: Visual explanation of PhyCell. [25]

The second layer is a 1×1 convolution representing the coefficients $c_{i,j}$.

After discretizing the continuous time PDE in Equation 1.3 (details in [25]), the prediction step can be written as:

$$\tilde{\mathbf{h}}_{t+1} = \mathbf{h}_t + \Phi(\mathbf{h}_t), \tag{1.6}$$

and the correction step is:

$$\mathbf{h}_{t+1} = \tilde{\mathbf{h}}_{t+1} + \mathbf{K}_t \odot (\mathbf{E}(\mathbf{u}_t) - \tilde{\mathbf{h}}_{t+1}). \tag{1.7}$$

Kalman gain $\mathbf{K}_t \in [0, 1]$ is approximated by learned convolution kernels $\mathbf{W}_h$, $\mathbf{W}_u$ and bias $\mathbf{b}_k$:

$$\mathbf{K}_t = \tanh(\mathbf{W}_h * \tilde{\mathbf{h}}_{t+1} + \mathbf{W}_u * \mathbf{u}_t + \mathbf{b}_k). \tag{1.8}$$

Note that if $\mathbf{K}_t = 0$, the input is not accounted for and the dynamics follows the physical predictor, and if $\mathbf{K}_t = 1$, the latent dynamics are reset and only driven by the input. This is similar to gating mechanisms in LSTMs. [25, 29]

Chapter 2

Related work

Forecasting precipitation with deep learning methods has seen notable progress in recent years. ConvLSTM and PhyDNet described previously are good examples. DeepMind's DGMR (Deep Generative Model of Radar) [30] is a more recent example. However, these models mostly work only with precipitation radar data. This chapter showcases some of the recent advances in using satellite data for precipitation forecasting, especially with machine learning methods.

2.1 MetNet

Machine learning precipitation forecasting models are mostly trained on very short lead times, usually not more than 3 hours. In 2020, Google Research developed a model called *MetNet* [1], and showed that it outperforms physics-based models up to a lead time of 8 hours (see Figure 2.3). It is relevant for this thesis because it uses satellite imagery as one of its input data.

MetNet is a neural network that forecasts rates of precipitation with a lead time of up to 8 hours, a spatial resolution of 1 km and a temporal resolution of 2 minutes. It covers a 7000 $\times$ 2500 km geographical area corresponding to the continental United States, shown in Figure 2.1.

The architecture (see Figure 2.2) uses axial self-attention [31] at its core to aggregate a large spatial context, which is necessary for a prediction for a lead time up to 8 hours. MetNet inputs a patch covering 1024 $\times$ 1024 km and predicts precipitation only for the center 64 $\times$ 64 km region of interest, leaving at least 480 km of spatial context on each of the four sides of the target patch.

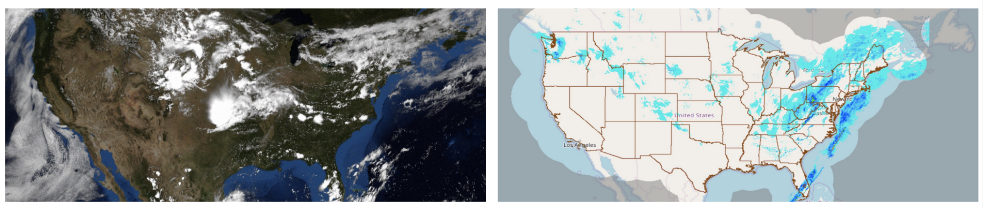


Figure 2.1: MetNet data source sample. Left: GOES-16 visual bands. Right: MRMS radar precipitation rates. [1]

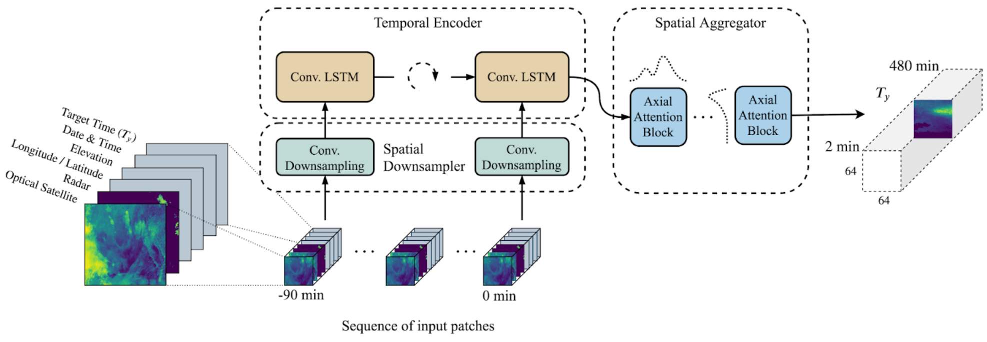


Figure 2.2: MetNet architecture. The input satellite and radar frames first pass through a spatial downsampler (convolutional and pooling layers) to reduce image size and memory consumption. They are then encoded by a convolutional LSTM over the 90 minutes of input data. Axial attention [31] layers enable the network to make use of the entirety of the encoded image to make a prediction for a given lead time T_y. [1]

As shown in Figure 2.2, the input includes a MRMS[5] radar image, 16 spectral bands of the GOES-16[6] satellite with a temporal resolution of 15 minutes up to 90 minutes to the past. Additional static features for the longitude, latitude and elevation of each location in the patch are included, as well as for the hour, day and month of the input time.

Although MetNet uses recurrent cells to encode the input, it does not produce output like a RNN. Instead, forward pass through MetNet makes a prediction for a single lead time. The desired lead time information is concatenated with the input features to inform the model. This enables the computation in MetNet to be conditioned on the lead time right from the beginning, allowing every aspect of the computation to be aware of it. With the ability to change the target lead time provided as input, the same MetNet model can be used to forecast for the entire range of target lead times it was trained on.

[5]`https://www.nssl.noaa.gov/projects/mrms/`
[6]`https://www.goes-r.gov/`

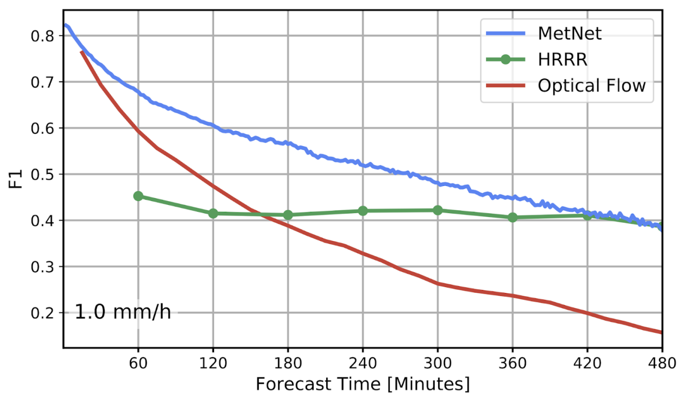


Figure 2.3: MetNet performance evaluated in terms of F1-score at 1.0 mm/h precipitation rate threshold. MetNet outperforms the optical flow method, as well as the physics-based model (HRRR) up to 8 hours ahead. [1]

The authors performed an ablation to study the importance of various input characteristics. As we can see in Figure 2.4, large spatial context is useful especially for long lead times, while large temporal context is not significant. The *MetNet-GOESOnly* configuration is especially interesting for this thesis, as it evaluates the ability of MetNet to predict precipitation rate from just the globally available GOES-16 satellite data. Despite starting off substantially worse, the performance of *MetNet-GOESOnly* approaches that of the full MetNet configuration with increasing hours of lead time, suggesting that input radar data becomes less necessary with time [1].

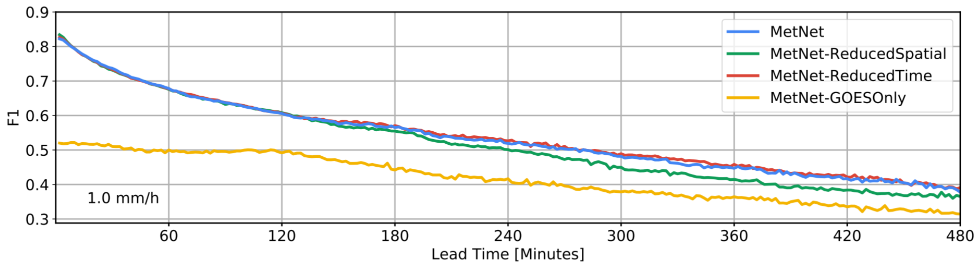


Figure 2.4: F1-score for each lead time of MetNet ablation experiments at the 1.0 mm/h precipitation rate threshold. *MetNet-ReducedSpatial* has a reduced input spatial context by half (512 km), *MetNet-ReducedTime* receives 30 minutes of input instead of 90, and *MetNet-GOESOnly* omits the MRMS radar input data. [1]

Although MetNet is considered a success, it is worth noting that it required up to 256 TPUs to train [1] and the exact dataset used is not public [32].

2.2 MetNet-2

MetNet-2 [33] is a successor to MetNet, and substantially improves on its performance (shown in Figure 2.6), extending the prediction to 12 hours. One of the main challenges to achieving such a long lead time is capturing a sufficient amount of spatial context in the input. MetNet-2 increases the input patch to 2048 $\times$ 2048 km, which is quadruple of what is used in MetNet. To be able to process this, MetNet-2 is distributed across 128 TPUs during training. It also employs changes in its architecture (also detailed in Figure 2.5). Attention layers are replaced with more computationally efficient convolutional layers. However standard convolutional layers have too small receptive fields to capture such a large context, so MetNet-2 uses dilated convolutions [34], and doubles their receptive field sizes layer after layer, to connect information which is spatially far apart in the input.

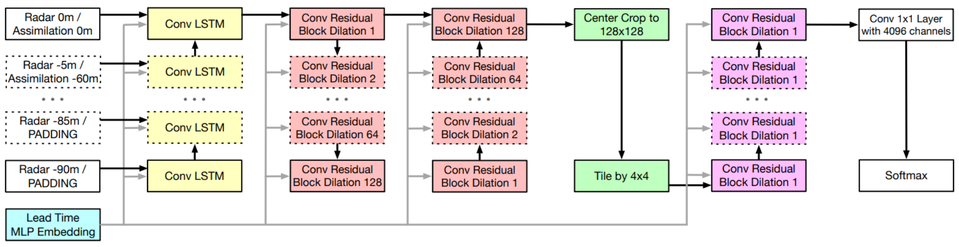

Radar 0m / Assimilation 0m
Radar -5m / Assimilation -60m
Radar -85m / PADDING
Radar -90m / PADDING
Lead Time MLP Embedding
Conv LSTM
Conv LSTM
Conv LSTM
Conv LSTM
Conv Residual Block Dilation 1
Conv Residual Block Dilation 2
Conv Residual Block Dilation 64
Conv Residual Block Dilation 128
Conv Residual Block Dilation 128
Conv Residual Block Dilation 64
Conv Residual Block Dilation 2
Conv Residual Block Dilation 1
Center Crop to 128x128
Tile by 4x4
Conv Residual Block Dilation 1
Conv Residual Block Dilation 1
Conv Residual Block Dilation 1
Conv Residual Block Dilation 1
Conv 1x1 Layer with 4096 channels
Softmax

Figure 2.5: MetNet-2 architecture. A convolutional LSTM embeds the input step by step. A stack of convolutional blocks with increasing dilation [34] captures the large context of the encoded input. After a center crop corresponding to the target patch area and a tiling operation that restores the 1×1 km spatial resolution, a final stack of convolutional blocks produces a distribution over precipitation levels for each target patch position. An embedding of the desired lead time conditions each convolutional layer of the network. [33]

Along with the radar and satellite images and other static data used in MetNet, MetNet-2 adds preprocessed starting state used in physics-based models as another source of input. This includes information such as temperature, humidity and wind direction, which is critical for longer forecasts up to 12 hours.

2.3 SEVIR dataset

To train deep learning weather forecasting models, large and diverse datasets containing high spatial resolution data are needed. Petabytes of weather data, for example from GOES (Geostationary Environmental Satellite System)[7] and NEXRAD (Next-Generation Radar)[8], are available to the public.

[7] `https://www.goes.noaa.gov/`

[8] `https://www.ncei.noaa.gov/products/radar/next-generation-weather-radar`

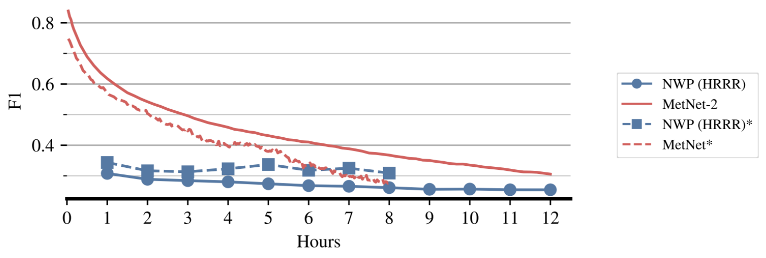


Figure 2.6: Comparison of F1-score using similar but distinct test sets of MetNet-2, MetNet, and NWP for both test sets. MetNet-2 outperforms MetNet substantially despite the data which MetNet-2 uses being harder for NWP. [33]

However, the size and complexity of these datasets is a hindrance to developing and training models. For predicting precipitation from satellite images specifically, we need both spatially and temporally aligned satellite and radar data.

To help address this problem, in 2020 the *SEVIR* (Storm EVent ImageRy) [35] dataset was created, which combines spatially and temporally aligned data from multiple sources, along with baseline implementations of deep learning models and evaluation metrics, to accelerate new machine learning innovations. It contains over 10 000 weather events from the continental United States, which each consist of 384×384 km image sequences spanning 4 hours. Samples in SEVIR include five different sensing modalities: three channels (C02, C09, C13) from the GOES-16 satellite, NEXRAD vertically integrated liquid (precipitation estimate from radar reflectivity) mosaics, and GOES-16 GLM (Geostationary Lightning Mapper) lightning flashes. See Figure 2.7 for an example. Events in SEVIR were carefully sampled to ensure the dataset contains relevant severe storm cases.

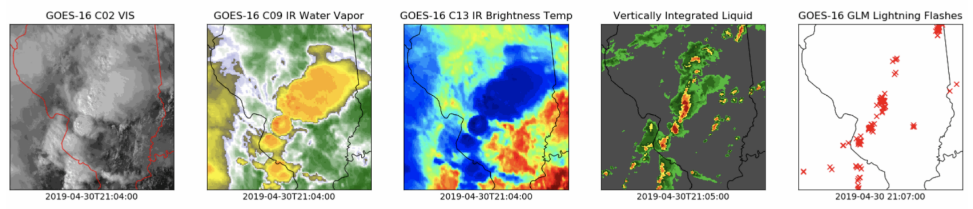

GOES-16 C02 VIS
GOES-16 C09 IR Water Vapor
GOES-16 C13 IR Brightness Temp
Vertically Integrated Liquid
GOES-16 GLM Lightning Flashes
2019-04-30T21:04:00
2019-04-30T21:04:00
2019-04-30T21:04:00
2019-04-30T21:05:00
2019-04-30 21:07:00

Figure 2.7: An example of spatiotemporally aligned images of different data types from the SEVIR dataset. [35]

Notable works utilizing the SEVIR dataset include [17] using GANs (Generative Adversarial Networks) [36] to make forecasts more realistic (less blurry), and [37] studying the effects of mixed precision training [38] in deep learning nowcasting models. However, according to [32] and our own research, SEVIR was not yet used for predicting precipitation from satellite data.

A similar purpose is shared by *Weather4cast 2022* [32], the dataset used in this thesis. It improves upon SEVIR with more satellite channels, a larger satellite context and a competition to predict precipitation from satellite data. More details in Chapter 3.

2.4 Synthetic radar

The use of geostationary satellite imagery is a tempting choice for precipitation nowcasting algorithms due to its ability to provide global and uninterrupted coverage. However, as the satellite does not directly observe the rain, a heuristic or machine learning algorithm must be used to extract the precipitation data. To fill in gaps outside the coverage of weather radar, many works attempt to develop a system that combines data from one or multiple non-radar sources to create a radar-like depiction of precipitation, sometimes referred to as *synthetic radar*. [39, 35]

2.4.1 Physics-based approaches

One approach to synthetic radar is the multi-sensor precipitation estimate (MPE) algorithm [40], which deduces a heuristic for precipitation detection from a physical model of the atmosphere, utilizing phenomena which are directed by well-known physical laws, such as absorption and scattering of light. However, the MPE algorithm is limited to convective rain and may produce incorrect results in areas with other forms of precipitation, such as frontal precipitation activities common in middle and high latitudes.

A more sophisticated version of a physics-based heuristic, the precipitation properties (PP) algorithm [41], combines input data of NWP models, physical properties of clouds, and satellite measurements. Radar observations are used to calibrate parameters of the algorithm. However, the retrieval of physical cloud properties, based on satellite observations at visible wavelengths, is limited to daylight hours.

2.4.2 Machine learning approaches

Simple machine-learning algorithms have also been used for precipitation detection from satellite imagery. Older techniques such as self-organizing feature maps are explored in [42, 43]. A study in [44] compared decision trees, shallow neural networks, and support vector machines for synthetic radar. However, the pixel-wise splits used in this work to obtain training and test sets may have led to overfitting, ignoring the smoothness of atmospheric phenomena in time and space. The study found that the best results were achieved in daytime conditions.

Another approach, utilizing only infrared and water vapor satellite spectral bands (available during night time) as input, is shown in [45]. It uses a fully-connected stacked denoising autoencoder [46] to reduce overfitting.

Authors of [39] apply a convolutional neural network to create synthetic radar. They fuse various input sources from satellite imagery, NWP and lightning flash detection. See example in Figure 2.8.

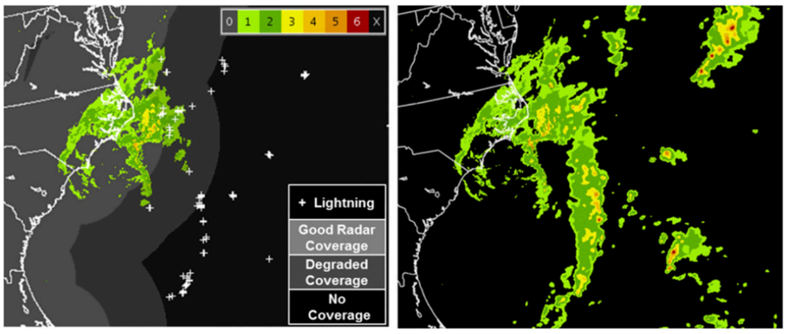


Figure 2.8: Left: NEXRAD radar depiction of precipitation intensity. Lightning flashes offshore (white plus symbols) indicate storm activity outside radar coverage. Right: The same analysis but with storms outside radar range filled in by fusing various non-radar sources using a CNN. [39]

2.4.2.1 U-Net

From the machine learning point of view, precipitation detection is similar to the problem of semantic segmentation, where the input is a multichannel image and the output is assigned to every pixel. This makes image-to-image neural networks like U-Net an ideal candidate for this task.

A notable example of using U-Net to create synthetic radar from satellite imagery is [2], illustrated in Figure 2.9 and Figure 2.10. To provide a better description of atmospheric condition to the model, NWP predictions are added as input. Specifically; convective precipitation rate, cloud work function, cloud water, precipitable water and convective potential energy on different levels, from the GFS model. Additionally, a topography map of the area, and the solar altitude of the current location and time are provided as input. Comparison with other approaches, including the MPE and PP algorithms mentioned earlier, is shown in Figure 2.11. Neural network (U-Net) and PP approaches produce better results during daylight, but U-Net performs well even at night and consistently beats the physics-based approaches.

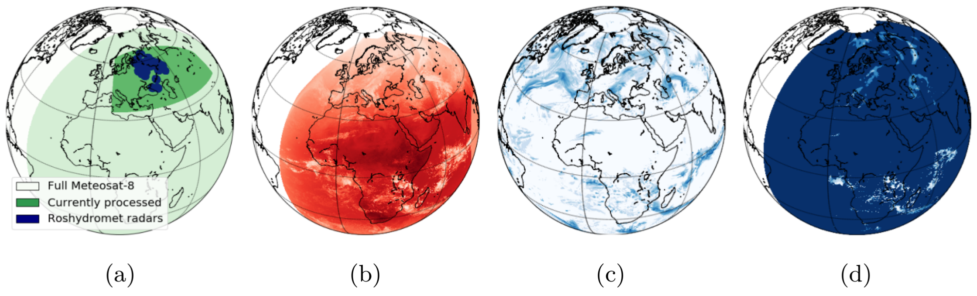


Figure 2.9: (a) The availability of data used in [2]: full view of the Meteosat-8 satellite, processed area inside it used as input, and radar coverage. (b) IR-097 (infrared channel) from Meteosat-8. (c) Total cloud water from the GFS model. (d) U-Net precipitation detection developed in [2].

This study also utilizes the synthetic radar for nowcasting. A neural network approach is compared with an optical flow algorithm. Although optical flow produced slightly better results in terms of F1-score in this work, it is stated that a neural network should surpass simple techniques if properly tuned. [2]

2.4.3 Global Precipitation Measurement

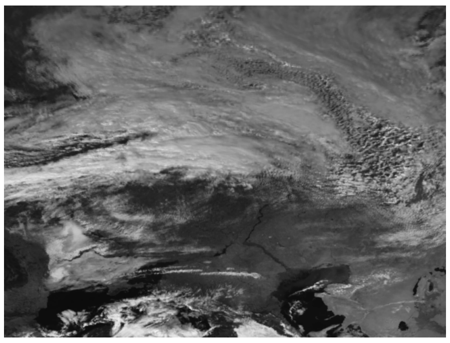
NASA's Global Precipitation Measurement (GPM) [47] mission is a collaborative effort with the Japan Aerospace Exploration Agency (JAXA) as well as other international space agencies to provide more accurate and frequent precipitation measurements worldwide.

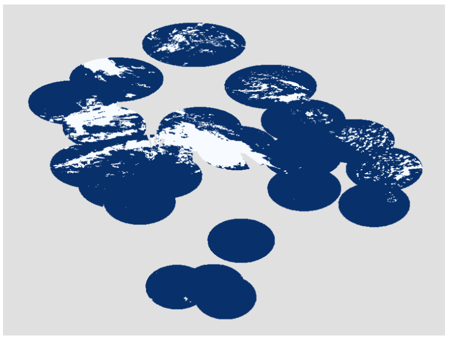
The GPM Core Observatory satellite, launched in 2014, carries the Dual-frequency Precipitation Radar (DPR) — a spaceborne active radar with two frequencies, and the GPM Microwave Imager (GMI) — a high-resolution multi-channel passive microwave (PMW) radiometer. GPM is assisted by a constellation of PMW satellites from other space agencies and missions. However, these satellites are on the low Earth orbit, and have limited range. The Core Observatory for example has a orbital period of 92.6 minutes. The constellation should provide retrievals on any given Earth coordinate 90 % of the time only roughly every 3 hours [47].

2.4.3.1 IMERG

One of the key products of the GPM mission is the Integrated Multi-satellitE Retrievals for GPM (IMERG) [48], which is a merged product of precipitation estimates from multiple satellite sensors, including GPM (radar and PMW), thermal infrared data from geostationary satellites, and also rain gauges around the world.

(a) Topographic map of the area.


(b) Input satellite view.


(c) Target ground truth radar. Colors denote rain (white), no rain (blue), no radar coverage (gray).

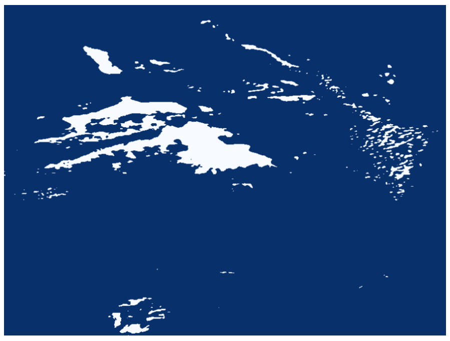

(d) U-Net radar estimation covering the entire satellite area.

Figure 2.10: Example of precipitation detection from satellite images. [2]

Estimates based on microwave and combined radar input data have higher quality due to the physically direct relationships between precipitation and the satellite data. Thermal infrared sensors estimates are lower quality, but they provide frequent interrupted coverage due to the geostationary orbit. The higher quality data are used as standard and are used as much as possible, the rest is calibrated to that standard and takes a secondary role. Finally, in locations where they exist, primarily land areas, monthly precipitation gauge data are used to control the biases that satellite data can exhibit. [49]

The IMERG algorithm is run three times, first 4 hours after the observation time (IMERG Early), then after 14 hours (IMERG Late), and finally 3.5 months later (IMERG Final), after all data including monthly rain gauge data are received. The Early product is less accurate but enables important low latency uses such as flood analysis, while the Late run is more complete and supports next-day work, such as crop forecasting. The Final product is considered the research-grade archival product intended for scientific analysis.

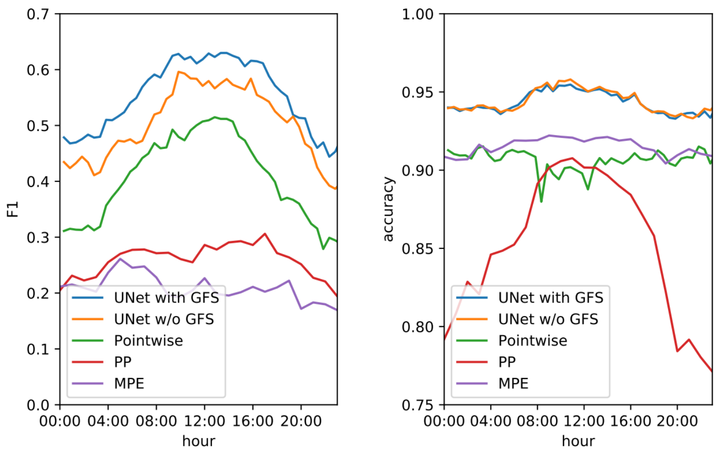


Figure 2.11: F1-score (left) and accuracy (right) of precipitation detection algorithms for each time of day. U-Net is tested both with and without GFS inputs. Pointwise refers to a simple CNN with only 1 × 1 convolutions, GFS not used. PP and MPE are physics-based approaches described in section 2.4.1. [2]

IMERG provides precipitation estimates with a 30-minute temporal resolution and a spatial resolution of approximately 10 km [49]. See example in Figure 2.12.

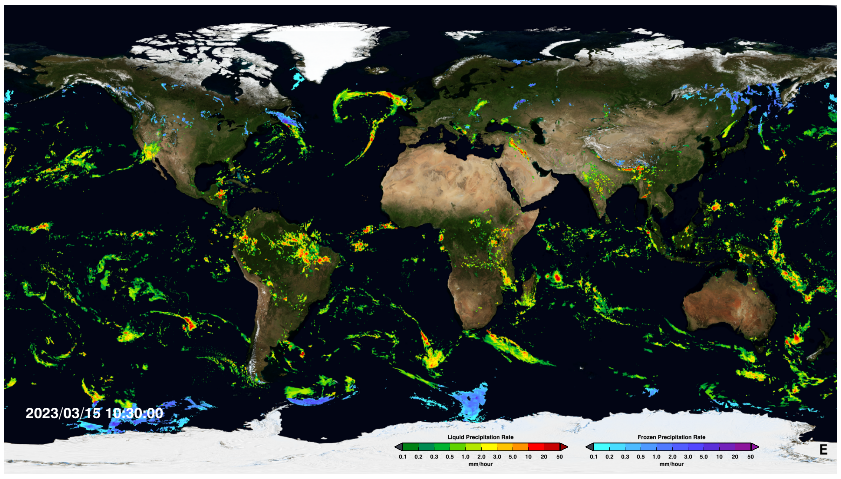


Figure 2.12: IMERG global precipitation estimation example. [49]

Because even the Early run has a latency up to 4 hours from observation time, associated with the acquisition and processing of satellite data, IMERG is not an ideal data source for nowcasting. The data archive can however be used to improve forecasting models. Notable machine learning works using IMERG include application of ConvLSTM in [50], comparison of 3D-UNet, ConvLSTM and other architectures in [51].

Chapter 3

Weather4cast 2022 competition

Weather4cast[9] is a NeurIPS competition organized by IARAI[10] (Institute of Advanced Research in Artificial Intelligence). The 2021 edition [52] dataset featured a range of weather products derived from the satellite data by EUMETSAT Satellite Application Facilities units dedicated to Nowcasting (NWC SAF),[11] from which temperature, tropopause turbulence probability and cloud mask were selected as target variables.

The aim of the 2022 edition was to predict future high resolution radar precipitation from lower resolution satellite images.

This competition was split into two stages. Stage 1 served as a warmup phase, allowing the participants to get familiar with the provided resources and rapidly experiment with a limited dataset. The entire training dataset was made public in Stage 2. Both stages had a test leaderboard, allowing participating teams to get familiar with the submission system and compare their submissions among each other and a baseline. To avoid overfitting over the course of the competition, the final competition results were determined from a different "heldout" leaderboard, which was open for only a few days at the end of Stage 2, along with a "heldout" part of the dataset.

The competition featured two challenges:

- the main **Core** challenge,
- and an additional **Transfer** challenge, which required a model to predict rainfall on significantly different data than it was trained on, spanning a different year and/or different geographical regions.

[9] `http://weather4cast.org/`
[10] `https://www.iarai.ac.at/`
[11] `https://www.nwcsaf.org/`

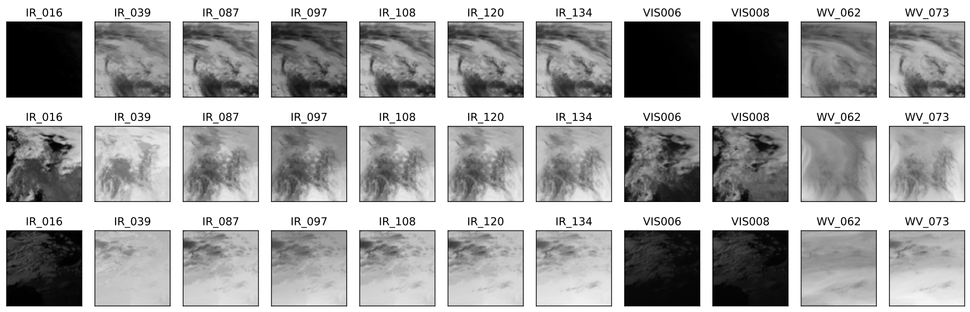


Figure 3.1: Satellite data examples. Rows are three random samples, columns are the available channels.

Table 3.1: Characteristics of the MSG SEVIRI satellite channels. [32]

Channel Name	Central Wavelength (μm)	Spectral Zone Characteristic	Type of Channel
`VIS006`	0.635	Solar Visible	Window (VIS)
`VIS008`	0.81	Solar Visible	Window (VIS)
`IR_016`	1.64	Solar Infrared	Window (VIS)
`IR_039`	3.90	Solar/Thermal Infrared	Window (VIS/IR)
`WV_062`	6.25	Thermal Infrared	H_2O Absorption (WV)
`WV_073`	7.35	Thermal Infrared	H_2O Absorption (WV)
`IR_087`	8.70	Thermal Infrared	Window (IR)
`IR_097`	9.66	Thermal Infrared	O_3 Absorption (IR)
`IR_108`	10.80	Thermal Infrared	Window (IR)
`IR_120`	12.00	Thermal Infrared	Window (IR)
`IR_134`	13.40	Thermal Infrared	CO_2 Absorption (IR)

3.1 Data

3.1.1 Satellite

Source of the satellite data is the MSG (Meteosat Second Generation) [53] geostationary meteorological series of satellites operated by the EUMETSAT[12] space agency. The satellites are equipped with an instrument called SEVIRI (Spinning Enhanced Visible Infra-Red Imager) that observes Earth in 12 spectral channels, including visible and near infrared (VIS), thermal infrared (IR), and a water vapor (WV) absorption band. 11 channels are available in this competition, see Figure 3.1 for examples. The spectral characteristics of each channel can be found in Table 3.1. Being geostationary, the satellite is located on the celestial equator plane at a fixed longitude and repeatedly captures images of the entire Earth disk from a constant perspective. Images are generated every 15 minutes in its nominal mode. The data provided in this competition is sourced from the satellite positioned at 0 degrees longitude.

[12] `https://www.eumetsat.int/`

3.1.2 Radar

In this competition, the “ground truth” precipitation is represented by weather radar data. Measuring precipitation is a common application of weather radar, which can provide the 3D structure of precipitation systems and track their movements over a relatively large area. A network of weather radar can cover an even larger domain. However, it is important to note that radar measurements are prone to errors caused by factors such as beam broadening, distance from the radar site, echoes from non-meteorological targets, terrain blockage, signal attenuation, and anomalous beam propagation. A comprehensive assessment of the pros and cons of weather radar for precipitation measurement, as well as an overview of current radar research, can be found in [54].

The radar data provided are 2D composites of the OPERA (Operational Programme for the Exchange of Weather Radar Information) [55, 56] from the EUMETNET[13] project.

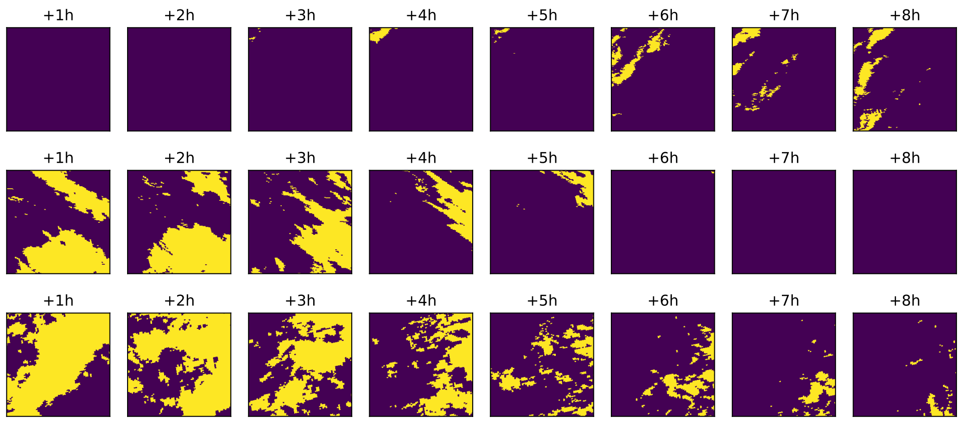


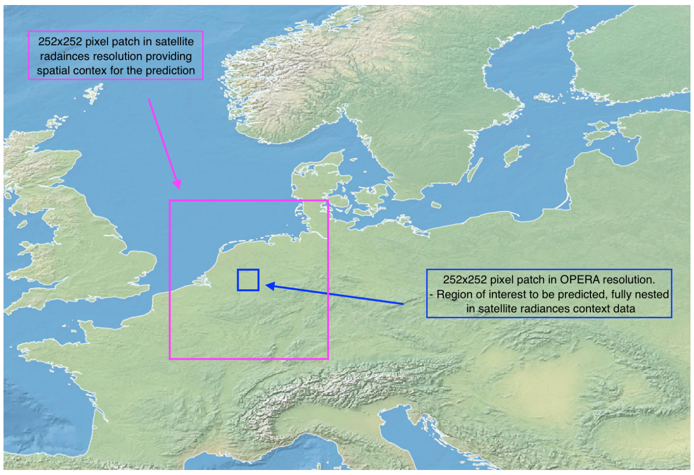
Figure 3.2: Target radar sequence examples. Rows are three random samples, columns are lead times up to 8 hours with a stride of 1 hour. Yellow color highlights rain, and purple means no rain (threshold 0.2 mm/h).

3.1.3 Competition-specific details

The OPERA radar network data and the MSG satellite data are in different geographical projections. Raw OPERA data are in Lambert Azimuthal Equal Area projection, which preserves the area with respect to the earth surface. The MSG data are in geostationary projection, where the spatial resolution is lower as a pixel gets further from the satellite. To ease the training of the models in this competition, the radar data have been converted to geostationary projection. With this, both satellite and radar frames align geographically, which is especially important for convolutional networks.

[13]`https://www.eumetnet.eu`

Both satellite and radar data are provided as image-like 2D arrays of 252×252 pixels. However, despite being the same size, they correspond to a different spatial area. While a satellite frame corresponds to an area of 3024×3024 km (12×12 km per pixel), a radar frame covers a 6 times smaller (504×504 km) area, at a 6 times higher resolution (2×2 km per pixel). It geographically aligns with the center 42×42 pixels of the satellite frame, which is referred to as the region of interest. The rest of the satellite frame can be interpreted as the surrounding spatial context. This is also explained in Figures 3.3 and 3.4.


252x252 pixel patch in satellite radainces resolution providing spatial contex for the prediction
252x252 pixel patch in OPERA resolution. - Region of interest to be predicted, fully nested in satellite radiances context data
Figure 3.3: OPERA and MSG contexts explanation. [32]

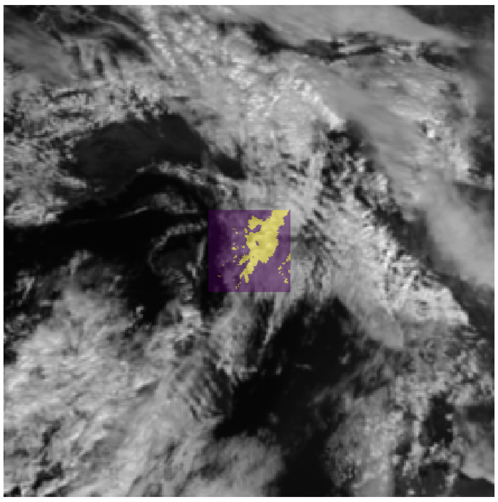


Figure 3.4: Binary radar image overlaid on top a VIS006 satellite channel, spatially and temporally aligned.

The data are geographically divided into several regions. Locations are shown in Figure 3.5. In Stage 1 of the competition, 3 regions spanning the year 2019 were available for training. In Stage 2, 4 more regions were made public as well as another year (2020) for all 7 regions. The Transfer challenge also features data from 3 different regions in the years 2019 and 2020, and one additional year (2021) for all 10 regions. See Figure 3.6 for a visual explanation.

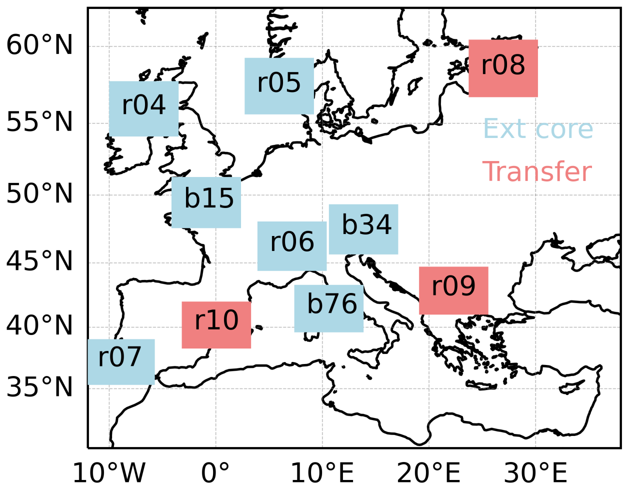


Figure 3.5: Geographical locations of each region. Core challenge regions are highlighted in blue, and additional Transfer challenge regions in red. [32]

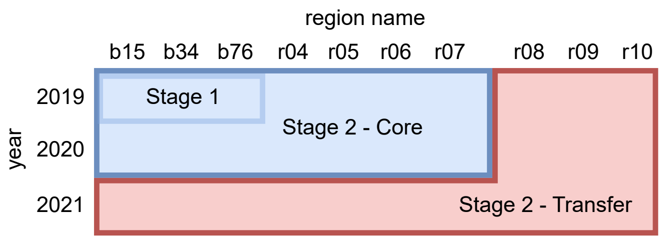


Figure 3.6: Availability of each region/year in the different stages and challenges. Note that the Transfer challenge data only contain test and heldout parts, which are meant for submissions, not for training. They contain only input (satellite) data, the target (radar) data are not public.

3.1.4 Static data

Static data for each region, including latitude, longitude, and topological height, were also provided, in the same resolution as satellite frames. See example in Figure 3.7.

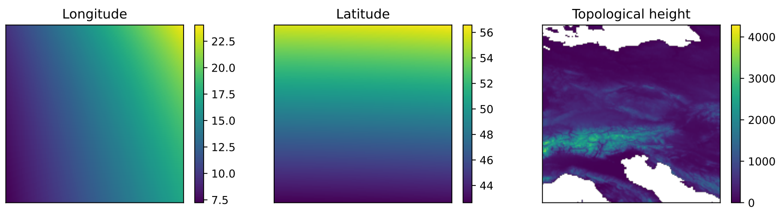


Figure 3.7: Static data for region b34 (boxi_0034). White color indicates missing values over bodies of water.

3.2 Task

The goal of this competition was to develop a model to predict radar up to 8 hours to the future from 1 hour of satellite input. Both sequences have a spatial resolution of 15 minutes, resulting in an input sequence of 4 frames and 32 output frames.

Although the provided radar frames contain numerical data representing rain rate (in mm/h), this competition's task was simplified to binary classification. This means a model should only predict 0/1 for rain/no rain for each pixel. In Stage 1, the binary threshold was specified as 0 mm/h. In Stage 2, it was increased to 0.2 mm/h, reasoned that having more training data should allow predicting rain events of higher sparsity. A higher threshold also helps with eliminating radar artifacts. However, this made the dataset even more imbalanced, resulting in a more challenging task. This difficulty increase is also reflected in lower evaluation metric values on the Stage 2 leaderboards compared to Stage 1.

As the task is binary classification, model performance can be measured with the commonly used binary classification metrics, such as accuracy, precision, recall, F1-score, etc. The main metric chosen in this competition, used for leaderboard evaluation, is IoU (Intersection over Union), also referred to as Jaccard similarity, Jaccard index, or CSI (Critical Success Index):

$$\mathrm{IoU}(Y, \hat{Y}) = \frac{|Y \cap \hat{Y}| + \epsilon}{|Y \cup \hat{Y}| + \epsilon},$$

where Y are ground truth positive pixels, $\hat{Y}$ are the predicted positive pixels, and ϵ is the smoothing parameter, preventing division by zero, $\epsilon = 1 \times 10^{-6}$ in this competition. This metric is appropriate for such an imbalanced dataset, as it only considers the positive pixels. For the leaderboard evaluation, IoU is computed independently for each region and year, then averaged.

3.3 Starter kit

Along with the dataset, organisers also provided a starter kit[14], which contains all the necessary code to train and evaluate a baseline 3D U-Net model out of the box:

- a PyTorch [57] data loader implementation,
- a PyTorch implementation of a 3D U-Net baseline model,
- training code using PyTorch-Lightning [58] library,
- implementation of several loss functions and metrics,
- and scripts to generate submissions from a trained model.

The dataset is provided out of the box split into train, validation, test, and heldout (final submission) parts, each in a separate file. The starter kit also contains a CSV file with timestamps of all data, which the provided data loader uses to correctly generate sequences. Training sequences are generated using a sliding window over the training split, yielding a total of 228 928 samples (for Stage 2 training data). Validation, test and heldout split are made of predefined non-overlapping sequences instead of a sliding window.

The provided data loader also does basic data preprocessing by default. Radar frames are converted to a binary mask for binary classification. Satellite data are standardized,

$$x' = \frac{x - \bar{x}}{\sigma},$$

where $\bar{x}$ and σ are the mean and standard deviation precomputed independently for each channel.

3.4 Solutions of other competitors

A submission to this competition also required publishing an extended scientific abstract. This section highlights the presented approaches of other participating teams in this competition, also summarized in [32]. These could also belong to Chapter 2, but as these solutions are specific to this competition, it is more appropriate here. Keep in mind that they were published *after* the competition, so they did not influence our solution.

Comparisons and results are presented later in Section 4.4.

[14]`https://github.com/iarai/weather4cast-2022`

3.4.1 Model Ensemble for Probabilistic Rain Prediction

Team *meteoai* present a solution [59] for probabilistic rain prediction using a model ensemble method from the baseline 3D U-Net and a space-time transformer, EarthFormer [60]. Instead of modifying model structure, the team focused on data preprocessing (cropping the input in half), training strategy (advanced loss functions), and post-processing (region-wise threshold optimization) to maximize the performance of the baseline models.

3.4.2 Vision Transformers for Weather4cast

The approach of team *team-name* [61] is based on Vision Transformers [62]. The team also introduces a set of configurations that can be applied to enhance results for various models as well as baseline-specific improvements.

The authors report that optimizing for Binary Cross Entropy is superior to other loss functions, and that a multiple model ensemble (a majority voting algorithm combining their top models) yields the most competitive results.

3.4.3 SIANet

Team *SI-Analytics* proposed SIANet (SImple baseline for weather forecasting using spatiotemporal context Aggregation Network) [63]. SIANet is an end-to-end model composed only of CNNs, similar to U-Net. In addition, SIANet has a different strategy from the general training strategies used by existing weather forecasting frameworks.

For the Transfer challenge, the team introduces a data augmentation strategy that considers wind direction, a smoother loss that considers spatio-temporal correlation, and a test-time geometric augmentation ensemble that performs inverse augmentation again during inference. [64]

3.4.4 RainUnet

Team *KAIST-CILAB* [65] presents a hierarchical U-shaped network, RainUnet, that utilizes a Temporal-wise Separable block. This block helps capture interframe correlations by decomposing the standard 3D convolution into spatial and temporal components, increasing the receptive field and enabling the network to learn long-range spatio-temporal dependencies.

The authors also experimented with various preprocessing strategies; filtering non-rainy sequences from the dataset, discarding some of the satellite channels, and cropping the input images.

3.4.5 Region-Conditioned Orthogonal 3D U-Net

Team *KAIST AI* [66] proposes a modified 3D U-Net architecture using region-conditioned layers, which take a one-hot encoded categorical input to generate a region-conditioned context that is added to the feature maps in the encoder block. The authors demonstrate that this module helps to ensure that the model is able to better distinguish between different regions in the input images and capture regional differences in the predictions.

Additionally, the authors make use of 1x1x1 orthogonal convolutions [67] and residual connections. To further improve the performance, the authors also apply several training strategies, including mixup, self-distillation, and feature-wise linear modulation (FiLM) [68].

Chapter 4

WeatherFusionNet

This chapter describes a novel approach we developed and submitted to the Weather4cast 2022 competition. We tackled the competition as part of an ongoing research project at the Data Science Laboratory (DataLab)[15] at FIT CTU. Although it was a team effort, the work presented in this thesis is my own contribution.

We decided to apply our good experience with PhyDNet and U-Net and combine them into a more complex architecture. We call the resulting model *WeatherFusionNet* as it fuses several different ways to process the satellite data. We briefly present the model in [69].

4.1 Architecture

This section describes the architecture of our model. We start from a baseline model, then we gradually add each of the components and improvements, explaining them one by one. The overall model is demonstrated at the end.

First, we need to select an appropriate baseline as the core of our model. Convolutional RNNs, such as PhyDNet, are commonly used for extrapolating spatio-temporal sequences. In this case however, the input and output sequences are very different. It is not only an extrapolation task, but also a domain translation and super-resolution task at the same time. RNNs cannot be applied here in a straightforward way, as it is unclear what to use as input during prediction steps. If we simply used the output from the previous time steps as per usual, the network would receive completely different inputs during the encoding steps and the prediction steps.

[15] `http://datalab.fit.cvut.cz/`

Furthermore, in this task the output sequence has a relatively large length of 32 frames, which makes it difficult to fit any large recurrent networks (such as PhyDNet) into memory, during training, with a sufficient batch size, on our available hardware.

4.1.1 U-Net

The starter kit features a 3D U-Net model as a baseline. This is an appropriate choice for this task, it does not have the limitations as RNNs. However, it makes it difficult to include additional input features, because of 3D convolutions. If we wanted to concatenate any extra images, they would need to have a temporal dimension, equivalent to the input sequence. We decided to try the regular U-Net as a baseline instead, as it does not have this problem.

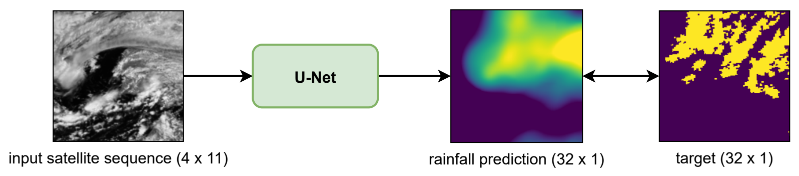

input satellite sequence (4 x 11)
U-Net
rainfall prediction (32 x 1)
target (32 x 1)

Figure 4.1: Illustration of the baseline U-Net training architecture. The numbers in parentheses denote the temporal and channel dimensions, respectively.

Interestingly, U-Net performed better than the provided 3D U-Net. It is also a less complex model and takes less time to train. For all of the U-Net modules in our architecture, we use the following version:

- hidden channel dimensions of 32, 64, 128, 256, 512 respectively,
- each 3×3 convolution is followed by a BatchNorm [11] layer and a ReLU,
- downscaling is realized by 2×2 max pooling with stride 2,
- and upscaling is realized by bilinear interpolation with scale factor of 2.

This U-Net has 44 input channels (flattened 4 input frames of 11 channels) and 32 output channels (the output sequence length). Note that the output length is a hyperparameter, which can be considered as a limitation.

Another limitation of U-Net is that it only works with spatial dimensions divisible by 32. This is because the contracting path of U-Net halves the resolution 5 times, $2^5 = 32$. To get around this, we pad the input images by 2 pixels on each side to increase the size to 256×256. The padding mode is set to *replicate* (repeating the outermost pixels), as this roughly extrapolates the images spatially. Then we crop the output of U-Net back to 252×252.

4.1.2 Crop & Upscale

Although this simple U-Net approach technically works, because the input and output has the same spatial dimensions, it ignores the fact that the output corresponds to a 6 times smaller geographical area. This goes for the provided 3D U-Net baseline as well.

We solve this with the *Crop & Upscale* method shown in Figure 4.2. We crop the center 42×42 pixels of all the 32 output frames of U-Net, corresponding to the region of interest. Then we upscale them to the target resolution.

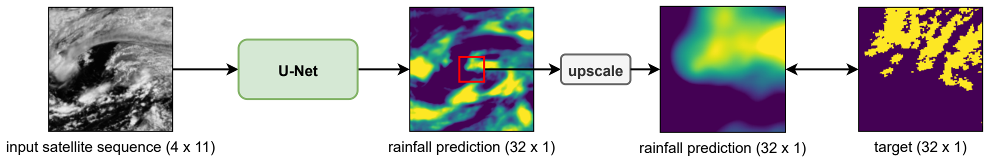

U-Net
upscale
input satellite sequence (4 x 11)
rainfall prediction (32 x 1)
rainfall prediction (32 x 1)
target (32 x 1)

Figure 4.2: Illustration of the U-Net Crop & Upscale training architecture.

It can be tempting to think that this is the same concept as the contracting and expanding mechanisms in U-Net. However, that is not the case. Although the hidden channels in U-Net have a smaller resolution, they still spatially correspond to the entire input. The residual connections in U-Net make this even more apparent. While the contracting and expanding mechanisms can propagate information across large distances, it has to be propagated in the same way spatially everywhere, because of the spatial invariance property of convolutions. With our approach, U-Net can make use of the entire surrounding satellite context, while everything still being spatially aligned. A similar forward pass cropping approach is also used in [33, 63].

The upscale operation is realized by bilinear interpolation, as in the expanding path of U-Net, with scale factor of 6. We experimented with more sophisticated approaches during Stage 1 of the competition, but we did not observe any improvement. Likely due to the fact that it is already very difficult to forecast accurately up to 8 hours in the satellite resolution, increasing the resolution hardly makes it any more accurate. But there is room for more research in this part.

As visible in Figure 4.2, the cropping causes U-Net to compute excess output outside of the region of interest. However this output does not participate in backpropagation, so the excess computation is minimal. It should be possible to modify U-Net to only compute the target region, but it is not worth the implementation effort.

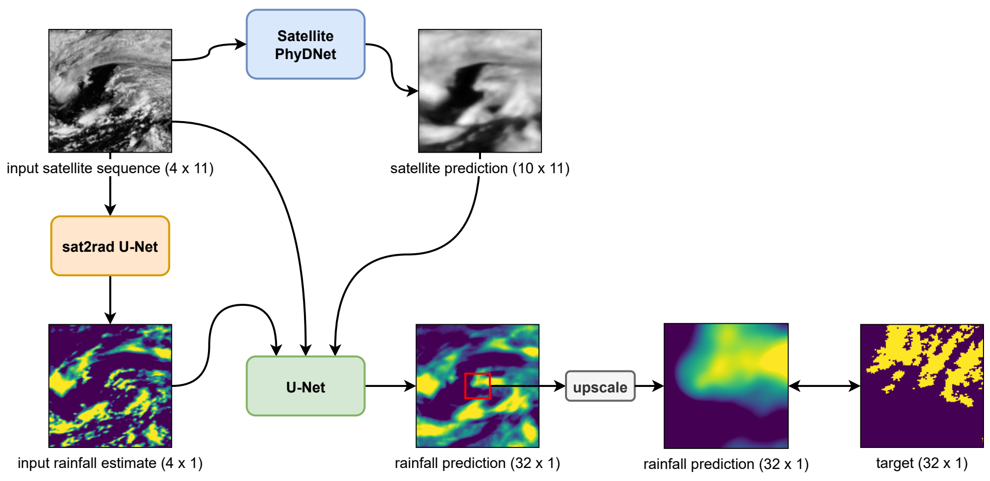

Satellite PhyDNet
input satellite sequence (4 x 11)
satellite prediction (10 x 11)
sat2rad U-Net
U-Net
upscale
input rainfall estimate (4 x 1)
rainfall prediction (32 x 1)
rainfall prediction (32 x 1)
target (32 x 1)

Figure 4.3: Illustration of the WeatherFusionNet training architecture.

4.1.3 Satellite PhyDNet

As explained previously, RNNs cannot be easily used to predict radar in this task, because we do not have the input radar sequence during inference. However, we can use them on the satellite data. Future satellite frames are obviously not available during inference, but they are actually present in the training data, and not used at all by the previously described end-to-end approaches.

We decided to train PhyDNet to essentially extend the input satellite sequence. This will be referred to as the *Satellite PhyDNet*. The predicted frames are then used as additional input features later. We limit the output sequence length to 10 frames, based on our memory limitations and past experience with PhyDNet.

4.1.4 Sat2Rad U-Net

In a similar fashion, we want to make use of the "input" radar sequence present in the training dataset. We train a module that we call *Sat2Rad* (satellite to radar). This network is trained to estimate the rainfall at the current time step of a single satellite frame. By training it this way, we believe it can efficiently extract information about the current rain situation in the input sequence, without having to predict the future. This module is realized by a U-Net with 11 input channels and 1 output channel. Notice that this is essentially a synthetic radar model, similar to [2].

During training, this module uses the same cropping and upscaling method described previously, because we only have the target data for the center region.

However, this time the excess output is actually useful. We take advantage of the spatial invariance property, to estimate the rainfall for the entire satellite area, even though it is only trained on the 6 times smaller area. The entire output is then used as another input feature.

4.1.5 WeatherFusionNet

Finally, as shown in Figure 4.3, the outputs from the two previously described modules are combined, along with the input sequence, and fused by a U-Net. The Sat2Rad module is applied to all 4 input satellite input frames independently, generating 4 channels in total. These channels, along with the input sequence and the Satellite PhyDNet output, are flattened and concatenated, yielding a total of $44 + 110 + 4 = 158$ channels. The fusion U-Net takes these as input. Other than that, it functions exactly the same as the *U-Net Crop & Upscale* model.

4.2 Training and results

This section describes the training procedures and hyperparameters of the trained networks, and presents the empirical results.

Like in the starter kit, models are implemented with PyTorch [57], trained and evaluated with PyTorch-Lightning [58]. Experiments were tracked using Weights and Biases [70], also used to plot the training charts. Matplotlib [71] is used to plot other charts and prediction visualizations.

The three parts of WeatherFusionNet are trained separately. Training it end-to-end would require too much memory, and also likely defeat the purpose of the extra two modules, as the extra data described previously would not be used.

All training procedures use early stopping and model checkpointing, restoring the best model after training is terminated by early stopping. No learning rate scheduling was used. All of the models have been trained using a single NVIDIA A100-40GB GPU.

4.2.1 Sat2Rad U-Net

As described previously, the Sat2Rad module was not trained on sequences, but on individual frames. To achieve this, we set the input and output sequence length to 1 in the data loader parameters. We also subtract 1 from the output samples indexes, making them temporally aligned with the input. No further modifications were required.

This U-Net was trained with a batch size of 32. This is more than for the other modules, to balance out the fact that in this case one sample is only one frame, not a sequence. Other hyperparameters (except the batch size) are listed in Table 4.2 and explained later in Section 4.2.3. Training metrics are shown in Figure 4.4, example predictions in Figure 4.5.

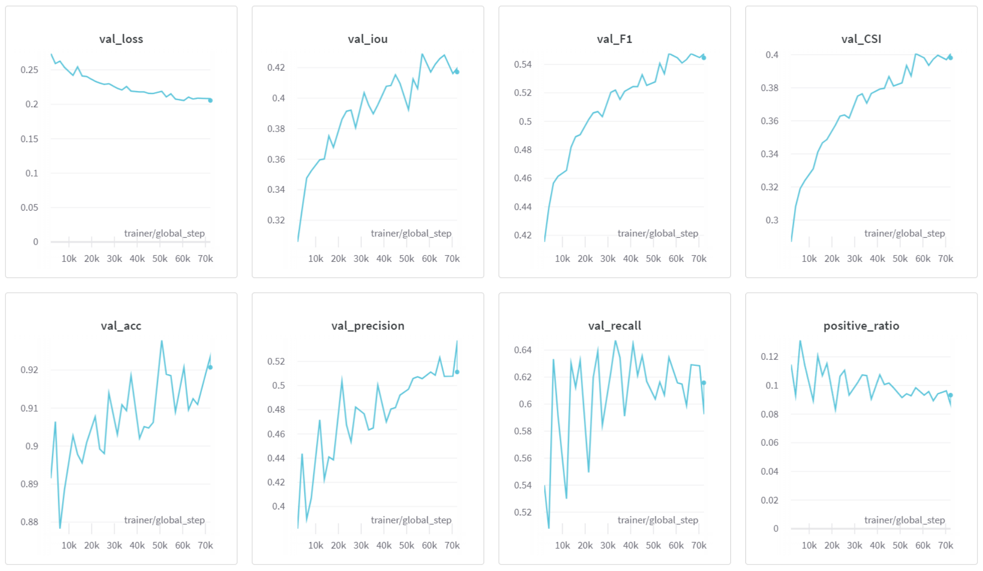


Figure 4.4: Sat2Rad U-Net evolution of validation metrics during training; loss (BCE), binary classification metrics (IoU, F1-score, CSI, accuracy, precision, recall) and positive ratio (how much percent of the output pixels are positive).

4.2.2 Satellite PhyDNet

The Satellite PhyDNet module is trained only using satellite data. We set the satellite sequence length to 14 in the data loader parameters and then split it into 4 input and 10 output frames.

Because the provided validation set was not designed for longer sequences, we used part of the training set as a validation split. Specifically, the first 150 000 samples were used for training, and the rest was used to generate non-overlapping validation sequences, using a sliding window with a stride of 32 samples.

Based on our past experience with PhyDNet, it was trained with a loss function L1L2 = L1+L2 = MAE+MSE, as well as the moment loss regularization. We used teacher forcing (using ground truth samples as input during prediction steps), starting with probability of 1, decreased by 5×10^{-5} every step. This is shown in Figure 4.6 along with other training metrics. Hyperparameters are listed in Table 4.1. We used a PyTorch-Lightning implementation of PhyDNet,

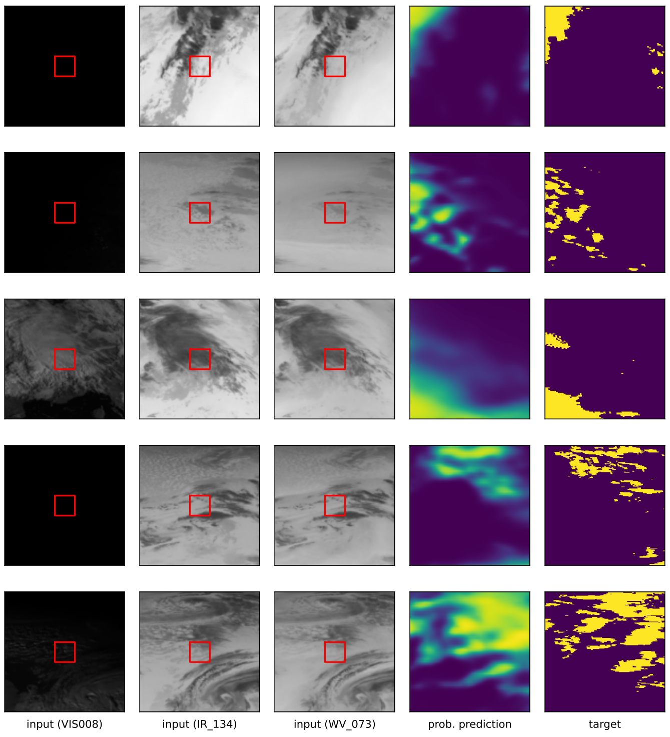


Figure 4.5: Sat2Rad U-Net example predictions. Each row is a different sample. First three columns are selected input channels. Red squares highlight the target area. Fourth column shows the raw probability output of Sat2Rad, before applying a threshold. Final column is the binary target, where yellow highlights rain, purple means no rain (0.2 mm/h threshold).

originally developed in [29]. As PhyDNet takes a relatively long time to train, we did not have time to experiment with different hyperparameters during the competition, we mostly used the default hyperparameters proposed in [25], except for the loss function. Figure 4.7 shows an example prediction.

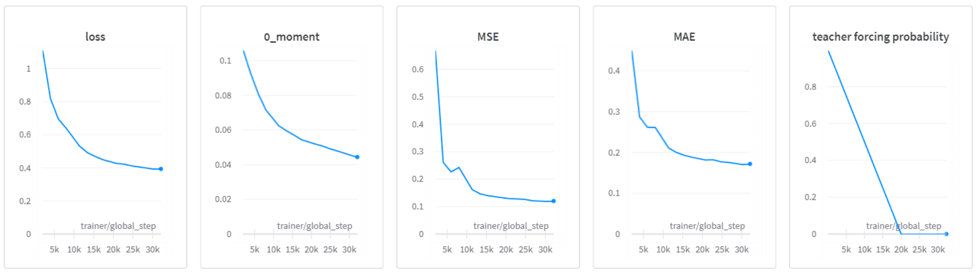


Figure 4.6: Satellite PhyDNet evolution of validation metrics during training; loss function and its components, and the teacher forcing probability.

Table 4.1: PhyDNet training hyperparameters

ConvLSTMCell	
Input dimension	64
Hidden dimensions	[128, 128, 64]
Kernel size	(3, 3)
PhyCell	
Input dimension	64
Hidden dimensions	[49]
Kernel size	(7, 7)
Batch size	16
Optimizer	Adam [72]
Learning rate	1×10^{-3}
Loss function	L1L2

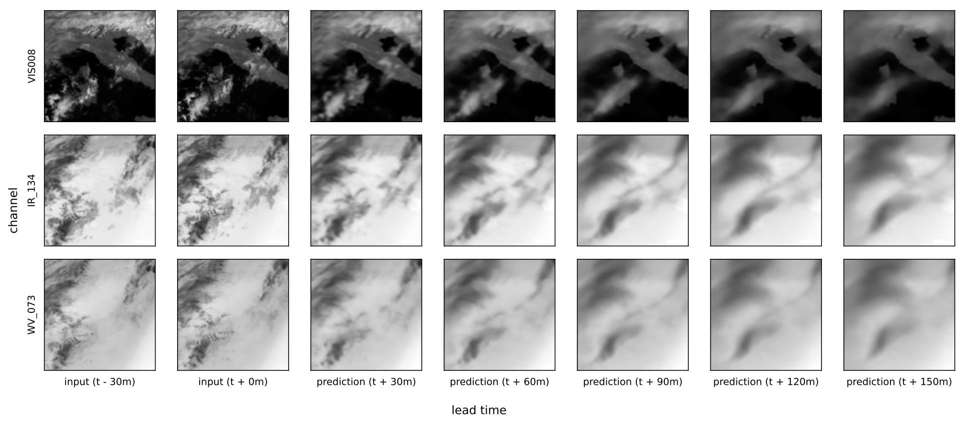


Figure 4.7: Satellite PhyDNet example prediction. Rows are three selected channels of both the input and prediction sequence, with a stride of 30 minutes.

4.2.3 WeatherFusionNet

The final U-Net model requires outputs from the previous two modules, but during training, they are frozen (their weights are not updated). Other than that, it is trained in the same way as all the other described U-Nets.

We experimented with different batch sizes, learning rates and loss functions (DiceLoss [73], FocalLoss [74], implemented in the starter kit) during Stage 1. Other hyperparameters were left as default from the starter kit. The best and final hyperparameters are listed in Table 4.2. Training charts are in Figure 4.9, example predictions in Figure 4.10, validation metrics in Table 4.3. IoU metric for each lead time is shown in Figure 4.8.

The positive weight hyperparameter in BCE (pixels with positive target are multiplied by this weight) helps to mitigate the dataset imbalance. The weight

Table 4.2: U-Net training hyperparameters

Hidden dimensions	[32, 64, 128, 256, 512]
Batch size	16
Optimizer	AdamW [75]
Learning rate	1×10^{-3}
Weight decay	1×10^{-2}
Loss function	Binary Cross Entropy
Positive weight	2.58

we used was provided in the starter kit, different for each competition stage. Another way to balance the network outputs is to change the binary output threshold. The default is 0 (or 0.5 after sigmoid). During Stage 1 we optimized the threshold on the validation set (model does not need to be trained again), which produced better IoU results. During Stage 2, the optimal threshold turned out to be the default one. This suggests the positive weight hyperparameter is also close to optimal.

Table 4.3: Validation metrics of the different models. Computed for each region and year, then averaged.

Model	**IoU**	**F1**	**Accuracy**	**Precision**	**Recall**
U-Net	0.2650	0.4153	0.9036	0.3834	0.4667
U-Net Crop & Upscale	0.2989	0.4553	**0.9146**	**0.4309**	0.4933
WeatherFusionNet	**0.3212**	**0.4828**	0.9128	0.4298	**0.5617**

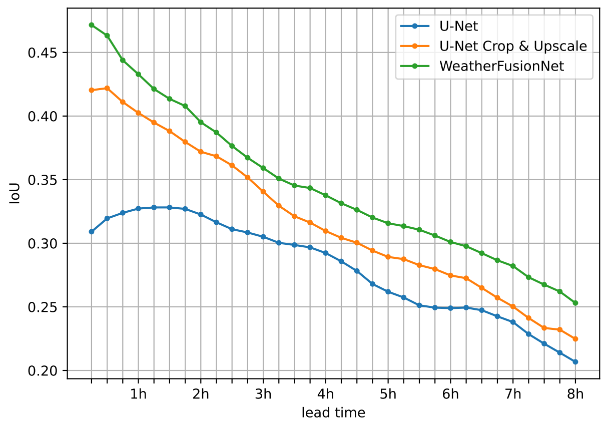


Figure 4.8: Comparison of models on the IoU metric, computed for each lead time. Interestingly, U-Net without cropping produces worse results for initial lead times. This is most likely caused by the spatially unaligned convolutions explained in Section 4.1.2.

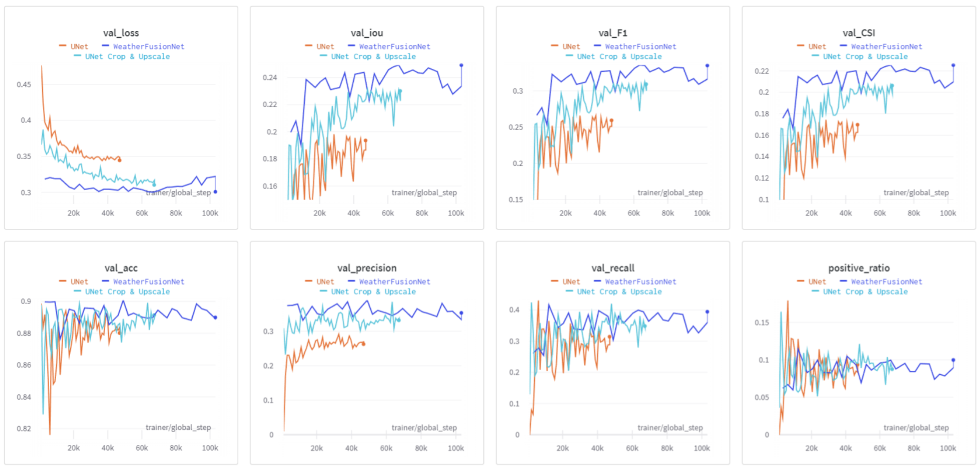


Figure 4.9: WeatherFusionNet (compared to the other described models) evolution of validation metrics during training; loss (BCE), binary classification metrics (IoU, F1-score, CSI, accuracy, precision, recall) and positive ratio (how much percent of the output pixels are positive). Computed on every batch and then averaged. The final value of each model (highlighted by a dot) is computed after restoring the best model checkpoint after early stopping.

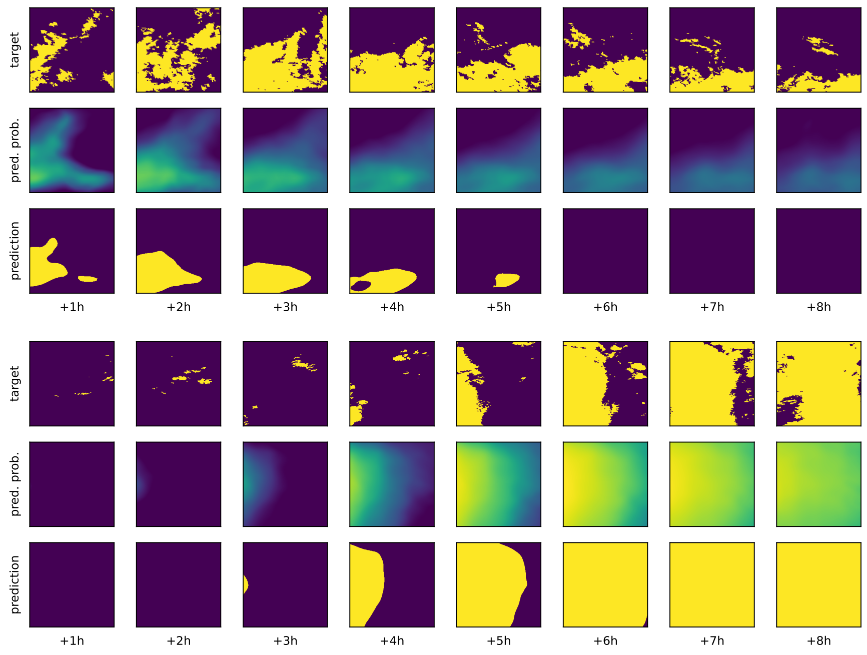


Figure 4.10: WeatherFusionNet two example predictions, lead time up to 8 hours, stride 1 hour. The rows represent; the binary target (0.2 mm/h threshold), the raw probability prediction of WeatherFusionNet, and the prediction after a 0.5 threshold.

4.3 Ablation study

In this section, we study the contributions of the sat2rad U-Net and Satellite PhyDNet modules. We experiment with two new models, each excluding one of the two modules. This does require retraining the fusion U-Net. Results are shown in Table 4.4 and Figure 4.11.

Interestingly, using only Satellite PhyDNet without Sat2Rad leads to worse results in terms of IoU. Sat2Rad contributes the most, especially for short lead times, which is to be expected. Adding Satellite PhyDNet seems to increase the performance on longer lead times (5 hours and more). This suggests that if one is only interested in short-term predictions, the Satellite PhyDNet can be excluded, and the model can be essentially reduced to only two U-Nets.

Table 4.4: Ablation study validation metrics. Computed for each region and year, then averaged. The first two columns denote whether Sat2rad U-Net or Satellite PhyDNet are used.

Sat2Rad	**PhyDNet**	**IoU**	**F1**	**Accuracy**	**Precision**	**Recall**
		0.2989	0.4553	0.9146	0.4309	0.4933
	✓	0.2955	0.4524	**0.9147**	**0.4401**	0.4782
✓		0.3149	0.4728	0.9140	0.4281	0.5354
✓	✓	**0.3212**	**0.4828**	0.9128	0.4298	**0.5617**

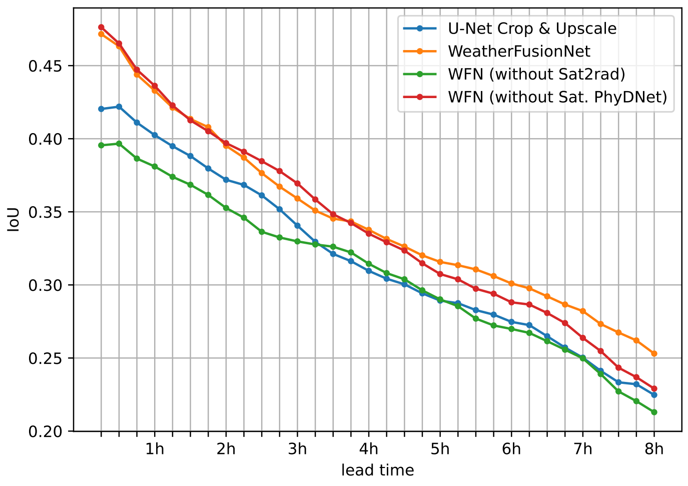


Figure 4.11: Comparison of ablation study models on the IoU metric, computed for each lead time.

4.4 Competition results

Table 4.5 presents the IoU metric of our models on the heldout sets. Only two models are shown because we were limited to three submissions, and one of them was used for another model, not presented in this thesis.

Table 4.5: IoU metric of our models on the heldout set from the final competition leaderboard. Computed for each region and year, then averaged.

Model	**Core Heldout IoU**	**Transfer Heldout IoU**
U-Net Crop & Upscale	0.2950	**0.2567**
WeatherFusionNet	**0.3162**	0.2488

WeatherFusionNet achieved better results on the Core heldout set, but performed worse than *U-Net Crop & Upscale* on the Transfer heldout set. We cannot analyze why is that the case, because the Transfer dataset targets are not public. But we can speculate that the more complex model could be slightly overfitted in terms of regions, leading to worse performance on the Transfer regions.

Table 4.6: Top ranked teams on the Stage 2 Core challenge and key features of their approaches (whether they are preprocessing inputs, employ an ensemble, use physics-based methods, or a transformer model). **ex-aequo* [32]

Rank	Team	avg IoU	Preprocess	Ensemble	Physics-based	Transformer
1	**FIT-CTU**	**0.316**	✓	×	✓	×
2	meteoai	0.307	✓	✓	×	✓
3*	SI Analytics	0.305	✓	×	✓	×
3*	TEAM-NAME	0.300	×	✓	×	✓
4	KAIST-CILAB	0.287	✓	×	×	×
5	KAIST-AI	0.274	✓	×	×	×
-	3D U-Net Baseline	0.254	×	×	×	×

As shown in Table 4.6, WeatherFusionNet achieved first place on the Core challenge. We also placed third in the Transfer challenge, both with WeatherFusionNet and *U-Net Crop & Upscale*.

We focused our efforts on the Core challenge, and submitted the same models for Transfer. The only team who developed a unique model for Transfer was *SI Analytics*, and they won the Transfer challenge. This shows the spatio-temporal shifts need special care, such as data augmentations, to achieve competitive results.

Chapter 5

More experiments

In this chapter, we present various additional experiments which were not part of our competition submission.

5.1 Static data

As mentioned in Section 3.1.4, the dataset also contains static data (latitude, longitude, topological height) for each region. We did not use it during the competition, because it was provided fairly late and there was no functionality in the provided data loader to handle the static data. We implemented it ourselves after the competition, modified WeatherFusionNet to use the static data, and show the results in this section.

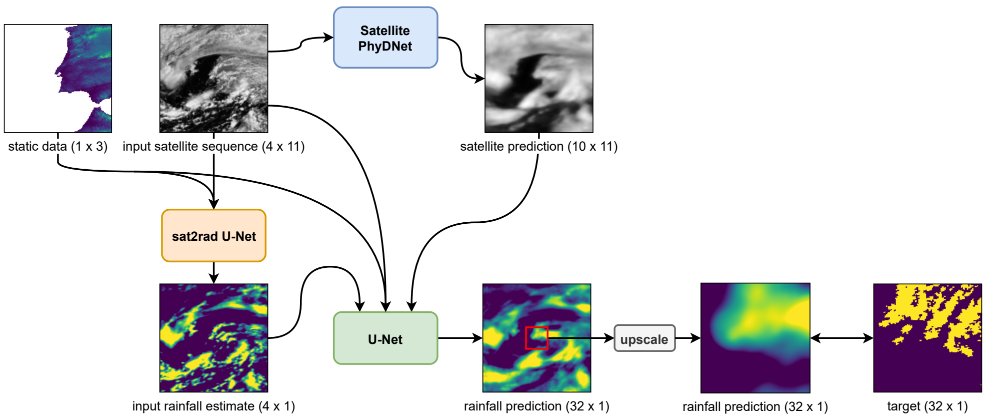

Satellite PhyDNet
static data (1 x 3)
input satellite sequence (4 x 11)
satellite prediction (10 x 11)
sat2rad U-Net
U-Net
upscale
input rainfall estimate (4 x 1)
rainfall prediction (32 x 1)
rainfall prediction (32 x 1)
target (32 x 1)

Figure 5.1: Illustration of the WeatherFusionNet training architecture, including static data (latitude, longitude, topological height).

We preprocess the static data to be usable by a neural network, by normalization and filling missing values (topological height values over bodies of water in this case) with zeros. As shown in Figure 5.1, we include the static data in the two U-Nets in our architecture. The only modification needed is adding 3 more input channels to both U-Nets, however the networks have to be re-trained. Inluding the static data to a recurrent network like PhyDNet is not a straightforward task, we leave this as subject for potential future research.

Table 5.1: Validation metrics of WeatherFusionNet with and without using static data. Computed for each region and year, then averaged.

Model	**IoU**	**F1**	**Accuracy**	**Precision**	**Recall**
WFN	0.3212	0.4828	0.9128	0.4298	0.5617
WFN (with static data)	**0.3272**	**0.4903**	**0.9134**	**0.4370**	**0.5653**

As shown in Table 5.1, including static data yields slightly better results. This is not surprising, it is reasonable to say local properties such as elevation and local climate influence precipitation. Some of this information can likely be inferred from the satellite frames, but providing this directly should help the network focus on the more important task — prediction.

5.2 Optical flow

Given the Sat2Rad U-Net module, we essentially have our own synthetic radar model. With this, we can test traditional radar nowcasting methods, such as optical flow extrapolation, and compare them with our deep learning model. This is similar to the approach developed in [2].

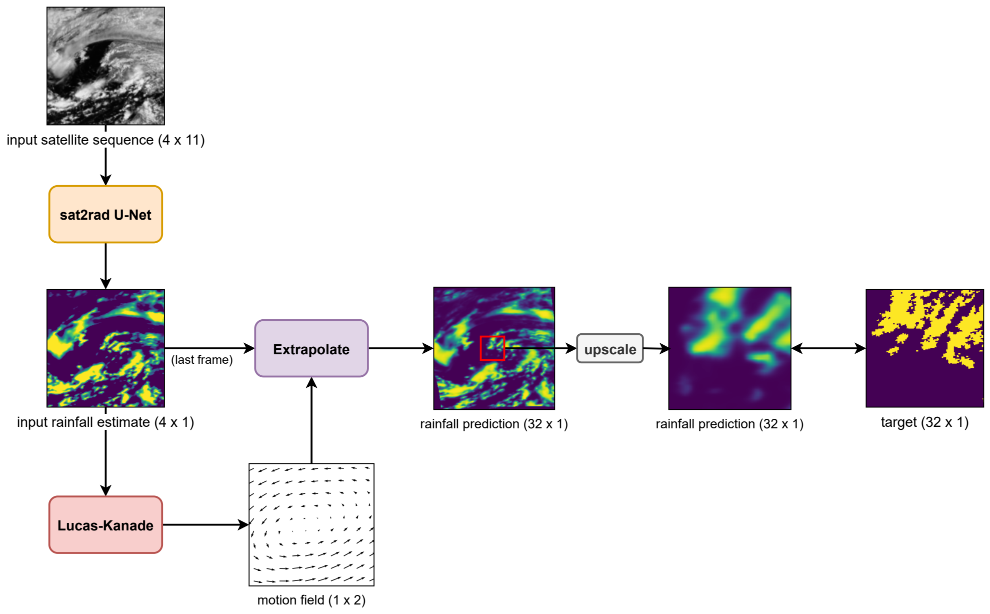

input satellite sequence (4 x 11)
sat2rad U-Net
input rainfall estimate (4 x 1)
(last frame)
Extrapolate
rainfall prediction (32 x 1)
upscale
rainfall prediction (32 x 1)
target (32 x 1)
Lucas-Kanade
motion field (1 x 2)

Figure 5.2: Illustration of the Sat2Rad Optical Flow method.

First, we introduce *Sat2Rad Optical Flow*, shown in Figure 5.2. Sat2Rad U-Net is applied on all 4 input satellite frames, generating 4 frames of synthetic radar. We use the `pySTEPS` [9] implementation of the Lucas-Kanade [5] method to estimate the optical flow of these frames, and then we use `pySTEPS` to extrapolate the last synthetic radar frame using the estimated motion field. This prediction is then cropped and upscaled as in the other models. Because the generated synthetic radar covers the entire satellite area, we can predict coming storms even if they are not within the target area initially.

Another common approach is the so called *persistence*. This simply repeats the last input frame for the entire prediction. This is obviously not a model suitable for real-world use, but it serves as a good baseline, testing if other prediction methods are not causing more harm than good. The *Sat2Rad Persistence* method applies Sat2Rad on the last satellite input frame and uses that for the entire prediction.

Results are shown in Table 5.2 and Figure 5.3. Optical flow and persistence method show comparable results for the initial lead times, but WeatherFusionNet significantly outperforms them in the long term.

Table 5.2: Validation metrics of the different models. Computed for each region and year, then averaged.

Model	**IoU**	**F1**	**Accuracy**	**Precision**	**Recall**
WeatherFusionNet	**0.3212**	**0.4828**	**0.9128**	**0.4298**	**0.5617**
Sat2Rad Optical Flow	0.2591	0.4045	0.9033	0.3701	0.4576
Sat2Rad Persistence	0.2200	0.3563	0.8939	0.3318	0.4012

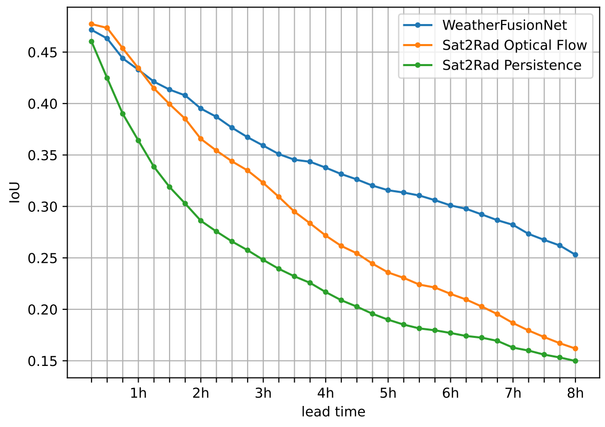


Figure 5.3: Comparison of several methods using Sat2Rad on the IoU metric, computed for each lead time.

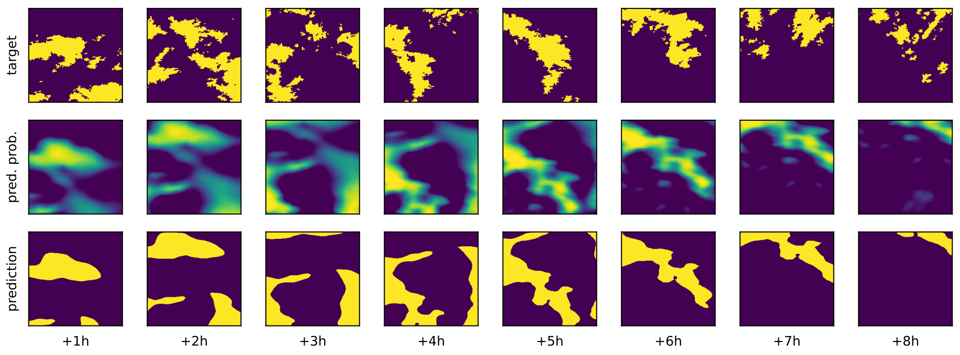


Figure 5.4: Sat2Rad Optical Flow example prediction, lead time up to 8 hours, stride 1 hour. The rows represent; the binary target (0.2 mm/h threshold), the raw probability prediction, and the prediction after a 0.5 threshold.

5.3 Radar to radar

When predicting radar from satellite, a question that may arise is; how much is the prediction worse than if we actually had input radar data? This experiment aims to answer that question, to some degree.

Fortunately, as explored in the previous chapter and used in the Sat2Rad module, the dataset actually does contain input radar data for train and validation sets. We can train a model to predict the target radar sequence from this input sequence. The important limitation is that this input corresponds to the same small area as the target, not the larger satellite area, missing the surrounding context. That is why this comparison is not completely fair. However, it does in a way reflect the real world, because radars have a limited range, compared to satellites.

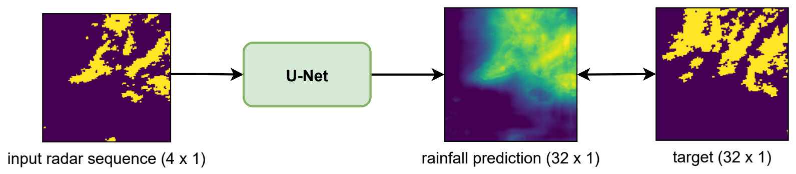


Figure 5.5: Illustration of the Radar2Radar U-Net training architecture.

Although we could now use more advanced models such as RNNs, to provide a more fair comparison we use a U-Net once again. As the input corresponds to the same area as the output, cropping and upscaling the prediction is no longer needed. We only need a U-Net with 4 input channels and 32 output channels. Other than that, it uses the same hyperparameters and training methods as the previous U-Nets. We refer to this model as *Radar2Radar U-Net*, illustrated in Figure 5.5.

The result is shown in Figure 5.6. As expected, the radar-to-radar model achieves substantially better performance for short lead times. However, it degrades quickly over time, and the satellite model convincingly outperforms it in the long run. This shows that the surrounding context is very important, and WeatherFusionNet is able to utilize it well.

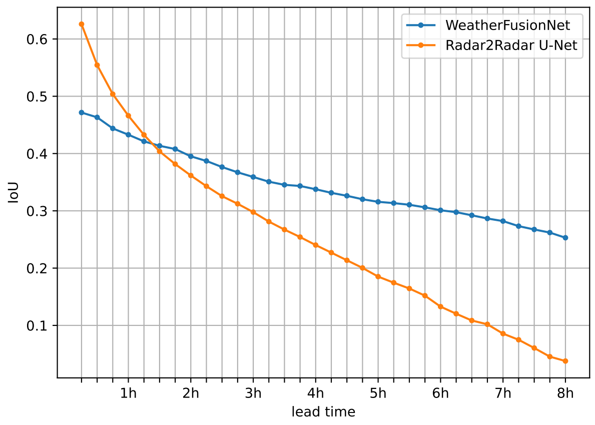


Figure 5.6: Radar2Radar U-Net in comparison with WeatherFusionNet on the IoU metric, computed for each lead time.

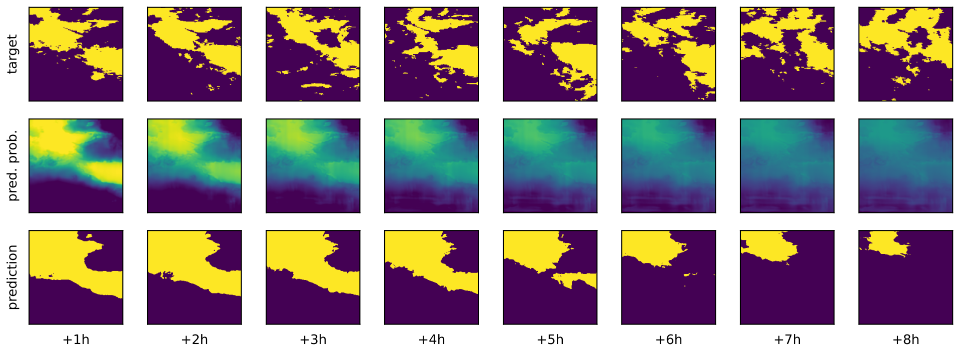


Figure 5.7: Radar2Radar U-Net example prediction, lead time up to 8 hours, stride 1 hour. The rows represent; the binary target (0.2 mm/h threshold), the raw probability prediction, and the prediction after a 0.5 threshold.

Conclusion

In this thesis, we researched existing methods for predicting rainfall from satellite data. We participated in the NeurIPS Weather4cast 2022 competition, where the challenge was to predict up to 8 hours of high resolution precipitation radar images from 1 hour of larger context but lower resolution multi-spectral satellite images.

We developed a novel deep learning model for this task, utilizing the U-Net and PhyDNet neural networks. We named it WeatherFusionNet, as it fuses three different ways to process the satellite data; predicting future satellite images, extracting rain information from the current frames, and using the input sequence directly. We also compared and demonstrated its performance over less complex baseline techniques. WeatherFusionNet proved its effectiveness by achieving first place in the Core challenge of the competition.

We further experimented with including static data (elevation, coordinates) in the input, leading to slightly better results. We showed that WeatherFusionNet outperforms a traditional optical flow extrapolation technique applied on a synthetic radar input. We also compared our satellite-to-radar model with a direct radar-to-radar model.

The last experiment shows that for longer-term predictions the large spatial context is more important than accurate local observations. This makes geostationary satellite data a reasonable source for forecasting in areas with little or no precipitation radar coverage. However, the numerical results are far from ideal, because forecasting up to 8 hours is already a very hard task to begin with, given the chaotic nature of weather.

Outline of future work

Weather forecasting remains a difficult challenge in general. As for our model specifically, enhancing the upscaling operation could improve performance, especially for short lead times. Another already mentioned potential improvement is modifying Satellite PhyDNet to use static data.

Many improvements could draw inspiration from the other solutions in the Weather4cast 2022 competition. Every team had a vastly different approach, yet managed to significantly improve upon the 3D U-Net baseline. Many of these ideas can be combined with ours. For example, the final U-Net in WeatherFusionNet can be replaced with more complex networks introduced by other teams. Various proposed training or data preprocessing techniques could also be utilized for our model.

More effort could also be put into the Transfer challenge, which is especially important for using these models in areas without radar coverage. Methods proposed by [64], specifically data geometric augmentation (flips, rotations), and a test-time augmentation ensemble, could be utilized to reduce overfitting. Region-wise cross-validation (using one or several regions only as a validation set) should be used to measure the performance on spatial shifts.

Another potential challenge is changing the task from classification to regression, where the target would be the exact rain rate instead of a binary class. The dimensions of the target are the same, so modifications to model architecture are not necessary. A regression loss function, such as MSE or MAE, needs to be used instead of BCE. However, the target data are still significantly imbalanced, as most of the time there is no rain at all. We can no longer use balancing techniques such as the positive weight paremeter in BCE. We could use data balancing methods such as under-sampling or over-sampling, or conduct more research for imbalanced regression techniques.

Bibliography

1. SØNDERBY, Casper Kaae et al. MetNet: A Neural Weather Model for Precipitation Forecasting. *arXiv preprint arXiv:2003.12140*. 2020.
2. LEBEDEV, Vadim et al. Precipitation nowcasting with satellite imagery. In: *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*. 2019, pp. 2680–2688.
3. BAUER, Peter; THORPE, Alan; BRUNET, Gilbert. The quiet revolution of numerical weather prediction. *Nature*. 2015, vol. 525, no. 7567, pp. 47–55.
4. KALNAY, Eugenia. *Atmospheric modeling, data assimilation and predictability*. Cambridge university press, 2003.
5. LUCAS, Bruce D; KANADE, Takeo. An iterative image registration technique with an application to stereo vision. In: *IJCAI'81: 7th international joint conference on Artificial intelligence*. 1981, vol. 2, pp. 674–679.
6. GUO, Shiqing; SUN, Nengli; PEI, Yanle; LI, Qian. 3D-UNet-LSTM: A Deep Learning-Based Radar Echo Extrapolation Model for Convective Nowcasting. *Remote Sensing*. 2023, vol. 15, no. 6, p. 1529.
7. AGRAWAL, Shreya et al. Machine learning for precipitation nowcasting from radar images. *arXiv preprint arXiv:1912.12132*. 2019.
8. AYZEL, Georgy; HEISTERMANN, Maik; WINTERRATH, Tanja. Optical flow models as an open benchmark for radar-based precipitation nowcasting (rainymotion v0. 1). *Geoscientific Model Development*. 2019, vol. 12, no. 4, pp. 1387–1402.
9. PULKKINEN, Seppo et al. Pysteps: an open-source Python library for probabilistic precipitation nowcasting (v1. 0). *Geoscientific Model Development*. 2019, vol. 12, no. 10, pp. 4185–4219.

10. RONNEBERGER, Olaf; FISCHER, Philipp; BROX, Thomas. U-Net: Convolutional networks for biomedical image segmentation. In: *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241.
11. IOFFE, Sergey; SZEGEDY, Christian. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: *International conference on machine learning*. PMLR, 2015, pp. 448–456.
12. TREBING, Kevin; STACZYK, Tomasz; MEHRKANOON, Siamak. SmaAt-UNet: Precipitation nowcasting using a small attention-UNet architecture. *Pattern Recognition Letters*. 2021, vol. 145, pp. 178–186.
13. ZHOU, Zongwei; RAHMAN SIDDIQUEE, Md Mahfuzur; TAJBAKHSH, Nima; LIANG, Jianming. UNet++: A nested U-Net architecture for medical image segmentation. In: *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support: 4th International Workshop, DLMIA 2018, and 8th International Workshop, ML-CDS 2018, Held in Conjunction with MICCAI 2018, Granada, Spain, September 20, 2018, Proceedings 4*. Springer, 2018, pp. 3–11.
14. ÇIÇEK, Özgün; ABDULKADIR, Ahmed; LIENKAMP, Soeren S; BROX, Thomas; RONNEBERGER, Olaf. 3D U-Net: learning dense volumetric segmentation from sparse annotation. In: *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2016: 19th International Conference, Athens, Greece, October 17-21, 2016, Proceedings, Part II 19*. Springer, 2016, pp. 424–432.
15. HO, Jonathan; JAIN, Ajay; ABBEEL, Pieter. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*. 2020, vol. 33, pp. 6840–6851.
16. HU, Xiaodan; NAIEL, Mohamed A; WONG, Alexander; LAMM, Mark; FIEGUTH, Paul. RUNet: A robust UNet architecture for image super-resolution. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*. 2019, pp. 0–0.
17. HU, Yuan; CHEN, Lei; WANG, Zhibin; PAN, Xiang; LI, Hao. Towards a more realistic and detailed deep-learning-based radar echo extrapolation method. *Remote Sensing*. 2021, vol. 14, no. 1, p. 24.
18. AYZEL, Georgy; SCHEFFER, Tobias; HEISTERMANN, Maik. RainNet v1. 0: a convolutional neural network for radar-based precipitation nowcasting. *Geoscientific Model Development*. 2020, vol. 13, no. 6, pp. 2631–2644.
19. WEYN, Jonathan A; DURRAN, Dale R; CARUANA, Rich. Improving data-driven global weather prediction using deep convolutional neural networks on a cubed sphere. *Journal of Advances in Modeling Earth Systems*. 2020, vol. 12, no. 9, e2020MS002109.

20. ZHOU, Kanghui; ZHENG, Yongguang; DONG, Wansheng; WANG, Tingbo. A deep learning network for cloud-to-ground lightning nowcasting with multisource data. *Journal of Atmospheric and Oceanic Technology*. 2020, vol. 37, no. 5, pp. 927–942.

21. SUN, Nengli; ZHOU, Zeming; LI, Qian; ZHOU, Xuan. Spatiotemporal Prediction of Monthly Sea Subsurface Temperature Fields Using a 3D U-Net-Based Model. *Remote Sensing*. 2022, vol. 14, no. 19, p. 4890.

22. HOCHREITER, Sepp; SCHMIDHUBER, Jürgen. Long short-term memory. *Neural computation*. 1997, vol. 9, no. 8, pp. 1735–1780.

23. SHI, Xingjian et al. Convolutional LSTM network: A machine learning approach for precipitation nowcasting. *arXiv preprint arXiv:1506.04214*. 2015.

24. OLAH, Christopher. *Understanding LSTM Networks* [online]. 2015-08. [visited on 2023-04-10]. Available from: `https://colah.github.io/posts/2015-08-Understanding-LSTMs/`.

25. GUEN, Vincent Le; THOME, Nicolas. Disentangling physical dynamics from unknown factors for unsupervised video prediction. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 11474–11484.

26. PIHRT, Jiří. *Spatio-temporal prediction using artificial neural networks*. 2021. Bachelor's Thesis. Czech Technical University in Prague, Faculty of Information Technology.

27. KALMAN, Rudolph Emil. A new approach to linear filtering and prediction problems. 1960.

28. LONG, Zichao; LU, Yiping; DONG, Bin. PDE-Net 2.0: Learning PDEs from data with a numeric-symbolic hybrid deep network. *Journal of Computational Physics*. 2019, vol. 399, p. 108925.

29. CHOMA, Matej. *Improving Deep Learning Precipitation Nowcasting by Using Prior Knowledge*. 2022. Master's Thesis. Czech Technical University in Prague, Faculty of Information Technology.

30. RAVURI, Suman et al. Skilful precipitation nowcasting using deep generative models of radar. *Nature*. 2021, vol. 597, no. 7878, pp. 672–677.

31. HO, Jonathan; KALCHBRENNER, Nal; WEISSENBORN, Dirk; SALIMANS, Tim. Axial attention in multidimensional transformers. *arXiv preprint arXiv:1912.12180*. 2019.

32. GRUCA, Aleksandra et. al. Weather4cast at NeurIPS 2022: Super-Resolution Rain Movie Prediction under Spatio-temporal Shifts. 2023. Submitted for publication.

33. ESPEHOLT, Lasse et al. Skillful Twelve Hour Precipitation Forecasts using Large Context Neural Networks. *arXiv preprint arXiv:2111.07470*. 2021.

34. YU, Fisher; KOLTUN, Vladlen. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*. 2015.

35. VEILLETTE, Mark; SAMSI, Siddharth; MATTIOLI, Chris. SEVIR: A storm event imagery dataset for deep learning applications in radar and satellite meteorology. *Advances in Neural Information Processing Systems*. 2020, vol. 33, pp. 22009–22019.

36. GOODFELLOW, Ian et al. Generative adversarial networks. *Communications of the ACM*. 2020, vol. 63, no. 11, pp. 139–144.

37. SAMSI, Siddharth; JONES, Michael; VEILLETTE, Mark M. Compute, time and energy characterization of encoder-decoder networks with automatic mixed precision training. In: *2020 IEEE High Performance Extreme Computing Conference (HPEC)*. IEEE, 2020, pp. 1–6.

38. MICIKEVICIUS, Paulius et al. Mixed precision training. *arXiv preprint arXiv:1710.03740*. 2017.

39. VEILLETTE, Mark S; HASSEY, Eric P; MATTIOLI, Christopher J; ISKENDERIAN, Haig; LAMEY, Patrick M. Creating synthetic radar imagery using convolutional neural networks. *Journal of Atmospheric and Oceanic Technology*. 2018, vol. 35, no. 12, pp. 2323–2338.

40. HEINEMANN, Thomas; LATANZIO, A; ROVEDA, Fausto. The Eumetsat multi-sensor precipitation estimate (MPE). In: *Second International Precipitation Working group (IPWG) Meeting*. 2002, pp. 23–27.

41. ROEBELING, RA; HOLLEMAN, I. SEVIRI rainfall retrieval and validation using weather radar observations. *Journal of Geophysical Research: Atmospheres*. 2009, vol. 114, no. D21.

42. HSU, Kuo-lin; GUPTA, Hoshin V; GAO, Xiaogang; SOROOSHIAN, Soroosh. Estimation of physical variables from multichannel remotely sensed imagery using a neural network: Application to rainfall estimation. *Water Resources Research*. 1999, vol. 35, no. 5, pp. 1605–1618.

43. BEHRANGI, Ali; HSU, Kuo-lin; IMAM, Bisher; SOROOSHIAN, Soroosh; KULIGOWSKI, Robert J. Evaluating the utility of multispectral information in delineating the areal extent of precipitation. *Journal of Hydrometeorology*. 2009, vol. 10, no. 3, pp. 684–700.

44. MEYER, Hanna; KÜHNLEIN, Meike; APPELHANS, Tim; NAUSS, Thomas. Comparison of four machine learning algorithms for their applicability in satellite-based optical rainfall retrievals. *Atmospheric research*. 2016, vol. 169, pp. 424–433.

45. TAO, Yumeng; GAO, Xiaogang; IHLER, Alexander; SOROOSHIAN, Soroosh; HSU, Kuolin. Precipitation identification with bispectral satellite information using deep learning approaches. *Journal of Hydrometeorology*. 2017, vol. 18, no. 5, pp. 1271–1283.

46. VINCENT, Pascal et al. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of machine learning research*. 2010, vol. 11, no. 12.

47. SMITH, Eric A et al. International global precipitation measurement (GPM) program and mission: An overview. *Measuring precipitation from space: EURAINSAT and the future*. 2007, pp. 611–653.

48. HUFFMAN, George J et al. NASA global precipitation measurement (GPM) integrated multi-satellite retrievals for GPM (IMERG). *Algorithm theoretical basis document (ATBD) version*. 2015, vol. 4, no. 26.

49. NASA. *Global Precipitation Measurement* [online]. [visited on 2023-03-15]. Available from: `https://gpm.nasa.gov/`.

50. GAMBOA-VILLAFRUELA, Carlos Javier; FERNÁNDEZ-ALVAREZ, José Carlos; MÁRQUEZ-MIJARES, Maykel; PÉREZ-ALARCÓN, Albenis; BATISTA-LEYVA, Alfo José. Convolutional lstm architecture for precipitation nowcasting using satellite data. *Environmental Sciences Proceedings*. 2021, vol. 8, no. 1, p. 33.

51. EHSANI, Mohammad Reza; ZAREI, Ariyan; GUPTA, Hoshin V; BARNARD, Kobus; BEHRANGI, Ali. Nowcasting-Nets: Deep neural network structures for precipitation nowcasting using IMERG. *arXiv preprint arXiv:2108.06868*. 2021.

52. HERRUZO, Pedro et al. High-resolution multi-channel weather forecasting–First insights on transfer learning from the Weather4cast Competitions 2021. In: *2021 IEEE International Conference on Big Data (Big Data)*. IEEE, 2021, pp. 5750–5757.

53. SCHMETZ, Johannes et al. An introduction to Meteosat second generation (MSG). *Bulletin of the American Meteorological Society*. 2002, vol. 83, no. 7, pp. 977–992.

54. SOKOL, Zbyněk et al. The role of weather radar in rainfall estimation and its application in meteorological and hydrological modelling—A review. *Remote Sensing*. 2021, vol. 13, no. 3, p. 351.

55. HUUSKONEN, Asko; SALTIKOFF, Elena; HOLLEMAN, Iwan. The operational weather radar network in Europe. *Bulletin of the American Meteorological Society*. 2014, vol. 95, no. 6, pp. 897–907.

56. SALTIKOFF, Elena et al. OPERA the radar project. *Atmosphere*. 2019, vol. 10, no. 6, p. 320.

57. PASZKE, Adam et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In: WALLACH, H. et al. (eds.). *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 8024–8035.

58. FALCON, William et al. *PyTorch Lightning*. 2019. Available also from: `https://www.pytorchlightning.ai`.

59. LI, Yang; DONG, Haiyu; FANG, Zuliang; WEYN, Jonathan; LUFERENKO, Pete. Super-resolution Probabilistic Rain Prediction from Satellite Data Using 3D U-Nets and EarthFormers. *arXiv preprint arXiv:2212.02998*. 2022.

60. GAO, Zhihan et al. Earthformer: Exploring space-time transformers for earth system forecasting. *Advances in Neural Information Processing Systems*. 2022, vol. 35, pp. 25390–25403.

61. BELOUSOV, Yury; POLEZHAEV, Sergey; PULFER, Brian. Solving the Weather4cast Challenge via Visual Transformers for 3D Images. *arXiv preprint arXiv:2212.02456*. 2022.

62. DOSOVITSKIY, Alexey et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*. 2020.

63. SEO, Minseok et al. Simple Baseline for Weather Forecasting Using Spatiotemporal Context Aggregation Network. *arXiv preprint arXiv:2212.02952*. 2022.

64. SEO, Minseok et al. Domain Generalization Strategy to Train Classifiers Robust to Spatial-Temporal Shift. *arXiv preprint arXiv:2212.02968*. 2022.

65. PARK, Jinyoung; SON, Minseok; CHO, Seungju; LEE, Inyoung; KIM, Changick. RainUNet for Super-Resolution Rain Movie Prediction under Spatio-temporal Shifts. *arXiv preprint arXiv:2212.04005*. 2022.

66. KIM, Taehyeon et al. Region-Conditioned Orthogonal 3D U-Net for Weather4Cast Competition. *arXiv preprint arXiv:2212.02059*. 2022.

67. WANG, Jiayun; CHEN, Yubei; CHAKRABORTY, Rudrasis; YU, Stella X. Orthogonal convolutional neural networks. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 11505–11515.

68. PEREZ, Ethan; STRUB, Florian; DE VRIES, Harm; DUMOULIN, Vincent; COURVILLE, Aaron. FiLM: Visual Reasoning with a General Conditioning Layer. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. 2018, vol. 32. No. 1.

69. PIHRT, Jiří; RAEVSKIY, Rudolf; ŠIMÁNEK, Petr; CHOMA, Matej. WeatherFusionNet: Predicting Precipitation from Satellite Data. *arXiv preprint arXiv:2211.16824*. 2022.

70. BIEWALD, Lukas. *Experiment Tracking with Weights and Biases*. 2020. Available also from: `https://www.wandb.com/`.

71. HUNTER, J. D. Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*. 2007, vol. 9, no. 3, pp. 90–95.

72. KINGMA, Diederik P; BA, Jimmy. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*. 2014.

73. SUDRE, Carole H; LI, Wenqi; VERCAUTEREN, Tom; OURSELIN, Sebastien; JORGE CARDOSO, M. Generalised dice overlap as a deep learning loss function for highly unbalanced segmentations. In: *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support: Third International Workshop, DLMIA 2017, and 7th International Workshop, ML-CDS 2017, Held in Conjunction with MICCAI 2017, Québec City, QC, Canada, September 14, Proceedings 3*. Springer, 2017, pp. 240–248.

74. LIN, Tsung-Yi; GOYAL, Priya; GIRSHICK, Ross; HE, Kaiming; DOLLÁR, Piotr. Focal loss for dense object detection. In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2980–2988.

75. LOSHCHILOV, Ilya; HUTTER, Frank. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*. 2017.

Appendix **A**

Acronyms

BCE	Binary Cross Entropy
CNN	Convolutional Neural Network
CSI	Critical Success Index
GPU	Graphics Processing Unit
IoU	Intersection over Union
LSTM	Long Short-Term Memory
MAE	Mean Absolute Error
MSE	Mean Squared Error
NWP	Numerical Weather Prediction
PDE	Partial Differential Equation
ReLU	Rectified Linear Unit
RNN	Recurrent Neural Network
TPU	Tensor Processing Unit
WFN	WeatherFusionNet

Appendix **B**

Contents of enclosed medium

`models`	model implementations
`thesis`	the thesis text directory
`thesis.pdf`	the thesis text in PDF format
`src`	the directory of LaTeX source codes of the thesis
`utils`	data loader and metrics implementations
`weights`	saved parameters of trained models
`environment.yml`	conda environment file
`predict-submission.py`	script to generate competition submission
`README.md`	instructions and description of the source code
`train*.py`	training scripts
`visualize*.ipynb`	visualization notebooks